STeP-1: A Set of Fundamental Tools for Persian Text Processing

Mehrnoush Shamsfard, Hoda Sadat Jafari, Mahdi Ilbeygi
NLP Research Laboratory, Faculty of Electrical and Computer Engineering, Shahid Beheshti University
Tehran, Iran
E-mail: m-shams@sbu.ac.ir, hodasj@yahoo.ca, ilbeygi_mahdi@yahoo.com

Abstract

Many NLP applications need fundamental tools to convert the input text into appropriate form or format and extract the primary linguistic knowledge of words and sentences. These tools perform segmentation of text into sentences, words and phrases, checking and correcting the spellings, doing lexical and morphological analysis, POS tagging and so on.
Persian is among languages with complex preprocessing tasks. Having different writing prescriptions, spacings between or within words, character codings and spellings are some of the difficulties and challenges in converting various texts into a standard one. The lack of fundamental text processing tools such as morphological analyser (especially for derivational morphology) and POS tagger is another problem in Persian text processing.
This paper introduces a set of fundamental tools for Persian text processing in STeP-1 package. STeP-1 (Standard Text Preparation for Persian language) performs a combination of tokenization, spell checking, morphological analysis and POS tagging. It also turns all Persian texts with different prescribed forms of writing to a series of tokens in the standard style introduced by Academy of Persian Language and Literature (APLL). Experimental results show high performance.

1. Introduction

There are many NLP primary tools developed for languages such as English to perform tokenization, morphological analysis, POS tagging, spell checking, chunking and so on. Processing texts for less resourced languages such as Persian which suffers from the lack of such tools, is more complicated and time consuming. All researchers who work on different aspects and levels of Persian text processing have to develop their own primary tools for a limited domain, task or application. There is no general wide-spread covering, available tool for doing primary NLP tasks for Persian.
This paper introduces a Persian text preprocessing toolkit called STeP-1. STeP-1 is the first step in processing Persian language written texts. It performs tokenization, morphological analysis, POS tagging and optional spell checking. Users can select arbitrary combination of these services in different depths for their own task and application.
In this paper after reviewing the problems and challenges of Persian text preprocessing, we will introduce STeP-1 as a solution. We will discuss tokenizer, morphological analyser and POS tagger as the main parts of STeP-1 in more details and show the experimental results.

2. Problems and challenges

Persian is among the languages with complex and challenging preprocessing tasks. Some of these challenges are listed below.
Imported letters from Arabic: Persian has 32 letters in its alphabet which cover 28 Arabic letters. In addition, there are some imported sounds such as ‘Tanwin’ and ‘Hamza” from Arabic which we use in some imported words in Persian. These words may be written in some different forms. For example ‘مسأله‘ ,’مسئله’ and ‘مساله’ are all forms of writing the word ‘problem’.
Unicode ambiguity: there are some letters such as ‘ی’ (i) and ‘ک’ (k) for which we have two Unicodes (one for Persian and one for Arabic). As some applications use the first and some the second one we have to unify their occurrences be-fore processing the text.
Different spellings: some words may be written with different letters such as ‘بلیت’ and ‘بلیط’ for ‘ticket’.
Different spacing: In Persian, Space is not a deterministic delimiter and boundary sign. It may appear within a word or between words. On the other hand there may be no space between two words. In these situations Persian will be similar to some Asian languages such as Chinese with no space between words. There are many words which can be written with space, short space or no space. For example ‘می¬رفت‘ ,’میرفت‘ ,’می رفت’ are all forms of ‘was going’.
Different writing prescriptions: APLL (2006) announces the rules and prescription for writing in Persian. Unfortunately these rules vary every few years and have a lot of exceptions. So NLP systems may receive texts with different styles. In some cases recognizing the correct style is not a straight forward task. Different prescriptions differ in the style of writing words, using or elimination of spaces within or between words, using various forms of characters and so on.
Transliterations: Writing foreign words (e.g. English) in Persian may result in some ambiguities in selection of letters. On the other hand as these words are not in the lexicons, tokenization and spell checking are not easy.
New words: Persian is a derivative and generative language in which many new words may be built by concatenating words and affixes. So the possibility of encountering a new word that is not available in the system’s lexicon is high.
Irregular and compound verbs: In Persian Verb constructions are mostly irregular. Many compound verbs can be derived from nouns and adjectives and in many cases the parts of these verbs have long distance dependencies.
Ezafe Construction: Ezafe construction is a special

construction in Persian which attaches nouns to their modifiers. Ezafe is a vowel which is pronounced but not written (in most cases). Non-written ezafe usually makes problems in chunking and syntactic and semantic processing of sentences while different forms of writing Ezafe makes ambiguities in tokenization and stemming.

3. Related Work

STeP-1 is a package that contains tokenizer, morphological analyzer and POS tagger. There is no other integrated package, so we refer to individual works on each part as related work.
For tokenizer part, the only main work is (Megerdoomian & Zajac, 2000) which was used as part of Shiraz project. This tokenizer has two phases. Splitting the input text into basic tokens is done in low-level tokenizer. The output of this phase goes to post-tokenizer for reattaching some specific suffixes. Our algorithm is more general and covers different type of acronyms and abbreviations, date, time, numbers and handles different styles of spellings. Also, in case of ambiguity, it shows all alternatives.
There are also different morphological analyzers that are proposed for Persian language. (Megerdoomian, 2004) uses a two-level morphological analyzer based on Xerox finite-state technology. In this work, if words like “dr” (in) or “v” (and) appears without space from their following words, they are considered as prefix. We take care of these kinds of attachments in tokenizer. Also we cover accusative pronominal clitics and many derivational affixes.
As another work we can mention Perstem (Dehdari, Lonsdale, 2008) written in perl language which was developed for inflectional morphology.
In compare with these morphological analyzers, our system has increased coverage and performance.
For Part of speech (POS) tagger part, we put an option in the system that user can choose between different POS taggers depends on percentage of unknown words in the input text. There are two POS tagger that are used in the system: TnT and Persian POS tagger. TnT (Brants, 2000) is a trainable POS tagger that uses trigrams and different approaches for handling unknown words. Persian POS tagger exploits a hybrid approach which is a combination of statistical and rule-based methods to tag Persian sentences (Fadaie & Shamsfard , 2009).

4. Solution: STeP-1

STeP-1 (Standard Text Preparation for Persian language) is designed to solve some of the above problems in an integrated package. In general it proposes the following activities for normalization and conversion of texts into a standard one.

1. Defining a computational standard script:
    a) Adding short-spaces between different parts of a word (or a compound word).
    b) Adding Spaces between words and phrases
    c) Introducing the spacing rules between punctuations, numbers and special cases (ex. date)
    d) Creating a lexicon with different spellings of words.
2. Converting texts to the standard script
    a) Looking up in a dictionary
    b) Checking the spelling
    c) Correcting the spacing
        i. replace white spaces with short spaces
        ii. Add white spaces (unknown words)

It then performs tokenization of the standard text and applies morphological analysis on the tokens to find the stems and tag the text by part of speech tags. Besides running a complete task for style correction and text preprocessing, user can choose each of the primary tasks in this toolkit to be ran separately.
In the rest of this section we will describe the tokenizer, morphological analyser and POS tagger as the three main modules of STeP-1.

4.1 Tokenization and Style Correction

Word Segmentation also known as tokenization focuses on recognizing word boundaries exploiting orthographic word boundary delimiters, punctuation marks, written forms of alphabet and affixes (Kiani & Shamsfard, 2008). Our proposed approach combines dictionary based and rule based methods and converts various prescribed forms of writing to a unique standard one. The developed tokenizer determines the words boundaries, recognizes multi part verbs, numbers, dates, abbreviations and some proper nouns. It employs a data base that contains 57284 entries including nouns, adjective, verbs, Prefixes and suffixes organized in different tables.
To describe the tokenization steps, let’s follow an example. Consider the following sentence[1]:

اوازشنبه(march13)ساعت۲:۳۰دقیقه،به مسافرت رفته است.

“OoAzShanbe(13march)Sa’at2:30Daghighe,beh mosaferat rafteh ast.”
(heFromSaturday(march13)hour2:30minute,to trip he is gone.)
‘he has gone to a trip from Saturday (march 13th) at 2:30.’
As it can be seen in this sentence some alphabetical strings (words) are concatenated to some other words or numerical strings. People can read this easily according to their knowledge about the language but computer should separate and tokenize the whole string to be able to process it. We thus expect the output to be as follows:
He | From | Saturday | (| march | 13|) | hour | 2:30 | minute|, | to | trip | HeIsGone |.
To create such results, the tokenizer goes through the following steps (in each case we show the string resulted by the step):

- Unify character coding: some Persian letters may be coded in Persian or Arabic unicodes. In such cases at the first step we convert all Arabic unicodes to their

[1] Throughout the paper we show the Persian examples with their pronunciations between double quotes (“), the word by word translations to English between parentheses and the translation of the whole sentence in italic between quotes (‘).

equivalent Persian ones.

- Split input text due to the position of white spaces and punctuation marks
- Separates numbers (integer or float), dates and English letters from their surrounding text by space. Numbers and English words should be written with a space with their surrounding word although this space can be deleted with-out making any ambiguity. For example in the above sample we will have:
  march13 → march | 13
  hour2 → hour | 2
  30minute → 30 | minute

- Adjust the spaces around punctuation marks. For this purpose, we should consider three different types of tags as follows:

a) startDelim: It is used for punctuation marks such as "(", "[" that should be written in the following pattern: <exp+space+startDelim+exp>

b) endDelim: It is used for punctuation marks such as ")", "]", "!" that should be written in this pattern: <exp+endDelim+space+exp>

c) betwDelim: It is used for punctuation marks such as "/", "-" that should be written like: <exp+betwDelim+exp>

For "." and ":" that can be written in different forms, first we set their tag to "endDelim". Later on depending on their surrounding words tags, we change their tag to "endDelim" or "betwDelim" if needed.
Other patterns used to adjust the spaces around the delimeters are following:

<number+":"+number>

<exp+":"+space+exp>

<abbreviation+"."+abbreviation>

<exp+"."+space+exp>

Applying this step on the above example will result in:
…Saturday(march| 13)hour |2:30| minute,to trip…
→…Saturday | (|march | 13|) |hour |2:30| minute|,| to…

- Recognize Verbs, nouns and adjectives. In this step we look the token in the lexicon. If it is not present there, then we should apply some stemming methods to remove affixes and check the stem in the lexicon

The person and number suffixes from verbs, plural signs from nouns and comparative markers from adjectives are some of the popular affixes to be removed.

- Recognize Abbreviation. Single letters in Persian (آ, ب…) or in English (A, B...) and name of English characters that are written in Persian (پی: pi, تی: ti…) are recognized as abbreviations.
- Recognize Numbers and English words. At this stage we just recognize and tag words written in English and other languages with similar alphabet (Latin based) but the module should be enhanced to recognize other foreign languages such as Arabic, Chinese, etc. we have introduced the tag 'englishWord' to tag these words and the tag 'number' for numerical strings. Recognize affixes and prepositions. For other words the program finds the correct tag for each word by looking at the data base. System uses 4 other tags which are shown in table 1.

Tag	Example
Postfix	est, er
Prefix	dis,un
verbPrefix	ing(coming before verb in Persian)
preposition	at, of

Table 1: 4 different tags that are used in the system

- Process undefined words

There may be still words not recognized by the previous modules. In these cases there are two alternative solutions; stemming and space insertion.

a) Stemming

The tokenizer just eliminates some specifics limited affixes. For more complex words the system uses the stemmer. The stemmer not only removes the affixes but also considers the orthographic changes during the inflection or derivation. For example in word "بیمهرگان" (invertebrata) the stemmer finds "مهره" (vertebra) as the stem by removing the negation affix ("بی"/ bi) and replace the letter "گ"(g) by "ه"(h) after removing the plural sign "ان"(An).

b) Space insertion

Some unknown strings may be converted to a sequence of known words by inserting spaces in appropriate places right in the boundary of words. for example strings like "ازتو" (from you) or "روزوشب" (day and night) in which no space is present, should be splitted from the word boundaries. In these cases we insert spaces after characters that just have a single form (alef(a),dal(d),zal(z),re(r),ze(z),zhe(j),vav(v))(Shamsfard, et al., 2009).
This is done by the segmenter. This function returns all possible forms of segmenting a phrase. Then each form goes to tokenizer again. Those forms that tokenizer can find tags for all their parts will be accepted. In some cases that we find more than one correct form, we show all possibilities in a square bracket. For example the string:

ازکاربرمیگشتیم

"AzKarBarMigashtim"
Can be splitted to

از کار برمی‌گشتیم [از کار بر می‌گشتیم، از کاربر می‌گشتیم]

Which mean either 'we were returning from work' or 'from user we were searching' or 'from work on we were searching'

- Rewrite the words with different spellings or styles into a unique, standard writing.

It changes 'ٔ' (Ezafe Construction) to "ی(y) + half space".

Table 2 shows some examples of the words with “hamza” that the program rewrites them into a standard writing.

Input	Output
مسؤولان	مسئولان
آمریکائی	آمریکایی
موثر	مؤثر

Table 2: Some examples of words with “hamza”

- Converts the spaces between parts of a word into a short space (or nothing) to concatenate separated parts of a single word.

For Compound words like “روز نامه” (news paper), the program writes its part with no space or with half space based on data base.

- Multi-Part Verb recognizer

For Verbs like “رفته است” (he/she is gone), that although we don’t need to write its parts with half space, we need to take them as one token. So the program returns both part as one token.

- Handling ambiguity

“می” (mi) can be verb prefix or noun (means wine). If the next word has tag verb and the verb doesn’t have any other prefixes, “mi” is considered as verb prefix otherwise we sets its tag to noun. Table 3 shows some examples.

Input	Output	explanation
جام می بیافتاد. “jam mi bioftad” ‘wine cup fall down.’	جام می بیافتاد. “jam mi bioftad”	“bioftad”→verb has prefix ‘bi’ “mi”→noun
علی را می دیدم. “Ali ra mi didam.” ‘I was seeing Ali.’	علی را می‌دیدم. “Ali ra mididam”	“didam”→verb “mi’→verb prefix

Table 3: Handling ambiguity in “می”

“ای” (I or A) can have four different meaning:
I) Name of English character ‘A’ that is written in Persian. If the previous word has noun tag and next word is dot, “ای” is considered as abbreviation. Table 4 shows an example.

Input	Output
کتاب ای . بی. سی book A .B. C	کتاب ای.بی.سی book A.B.C

Table 4: “ای” as abbreviation in the text

II) Suffix denoting second person
If the previous word has verb tag and it is ended with ‘ه’(h) and the next word is noun, “ای” can be written with half space or one space with previous word. Table 5 shows an example.

Input	Output
کجا رفته ای بچه ! “koja rafte I bache ! ‘where he is gone hey kid!’ or “koja rafteh A bache!” ‘where have you gone kid!’	کجا رفته‌ای [رفته ای] بچه! “koja rafteI [rafteh A] bache!”

Table 5: “ای” as verb prefix or interjection

III) Interjection
If the previous word has verb tag and it is not ended with ‘ه’(h) and the next word is not dot, “ای” is written with one space with previous word. An example is represented in table 6.

Input	Output
چه میخوری ای پسر ؟ ! “che mikhori A pesar ? !” ‘what are you eating o ye boy?’	چه می‌خوری ای پسر؟! “che mikhori A pesar?!”

Table 6: “ای” as interjection

IV) Indefinite article
If the previous word is noun that ended with ‘ه’ (h), and next word is not dot, “ای” is written with half space from previous word.

Input	Output
خانه ای خریدم. “khaneh I kharidam.” ‘I bought a house.’	خانه‌ای خریدم. “khanehI kharidam.”

Table 7: “ای” as indefinite article

“و” (v: and) can be abbreviation or preposition.
If the next or pervious word is dot it is considered as an abbreviation otherwise it is preposition.

In case of ambiguity in segmenter function, it shows all alternative tokenization solutions. Table 5 shows an example.

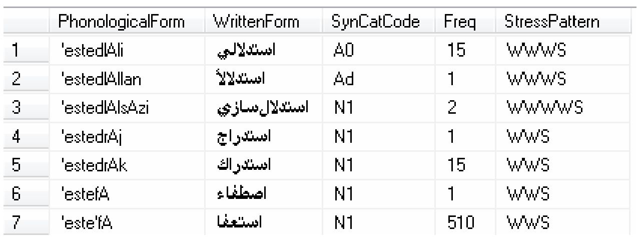
Input	Output
دربرابرباد AgainstTheWind	در بر ابر باد [در برابر باد] to against cloud wind [against wind]

Table 8: different alternative in “دربرابرباد”

4.2 Morphological Analyzer

The morphological analysis subsystem of STeP-1 extracts stems and affixes using inflectional and some derivational morphological rules in Persian. This analysis uses POS tags of the stem and the whole word to find the best stemming path. The algorithm uses a Finite State Automation (FSA), a data base consisting of words and

their Part Of Speech (POS) tags, a structure for keeping stem and its expected tags, two lists for keeping prefixes and postfixes which were eliminated from word and some rules for morphological analysis.
The data base contains 54347 Entries (Eslami, et al., 2004). Figure 1 shows part of this data base.



	PhonologicalForm	WrittenForm	SynCatCode	Freq	StressPattern
1	'estedlAli	استدلالي	A0	15	WWWS
2	'estedlAllan	استدلالاً	Ad	1	WWWS
3	'estedlAlsAzi	استدلال‌سازي	N1	2	WWWWS
4	'estedrAj	استدراج	N1	1	WWS
5	'estedrAk	استدراك	N1	15	WWS
6	'estefA	اصطفاء	N1	1	WWS
7	'este'fA	استعفا	N1	510	WWS

Figure 1: part of data base

The data base contains 33 different tags (Eslami, et al., 2004) that are used in stemmer. Table 9 shows some of the tags.

Tag	Description
A0	Adjective
Ab	Abbreviation
Ad	Adverb
N1	Noun
V1	Present verb
V2	Past verb
VPr	Verb prefix

Table 9: some of the tags that are used in stemmer

There are two FSAs, one for recognizing prefixes in the word and the other for recognizing suffixes.
Figure 2 shows part of postfix FSA. “ha” is plural sign, “shv” is verb and “y” is indefinite article.

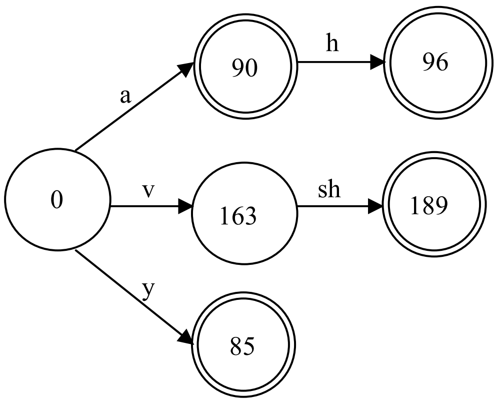


Figure 2: part of postfix FSA

After eliminating each affix from the word if we find the remaining in the lexicon with the expected POS tag, we accept it as a stem. Like, in word “تميز تر” (cleaner). When “تر” (er) is eliminated, the remaining part “تميز” (clean) has an adjective tag; therefore it is accepted as a valid stem. But in the word “كتاب تر” (booker), after eliminating “تر” (er), as “كتاب” (book) has noun tag, we don’t accept it as a stem.
Sometime we need to eliminate more than one affix to find the stem. In these cases, the system keeps the expecting tags for resulting words in a list. At the end, if the stem had one of the tags in the list, we accept this stem as a valid stem. This technique is used to prevent producing invalid stems for words.
There are special cases that even when the input is correct, the stemmer can not find any stem for the word. Like in the compound word “دانش آموزان” (students) that after eliminating “ان” (an: plural sign), program can not find the remaining, “دانش آموز” (student) in the data base. This is because there is space between two parts of the component word and we have this word with half space in the data base. In these cases we replace the space with half space and send the word to stemmer again.Table 3 shows some rules that are used in the morphological analyzer. Words in square bracket are optional.

Rule	Example
past person identifier + past root + mi + [accusative pronominal clitics]	می خوردم “mi khordam” ‘I was eating’
past root + mi + past person identifier + dasht	داشت می رفت “dasht mi raft” ‘he/she was going’
noun + plural sign	كتاب ها “ketab ha” ‘books'

Table 10: some rules in morphological analyzer

Checking all possible styles and forms of writing before applying the stemmer or morphological analyzer is essential to prevent mismatches.
Table 11 shows some examples of morphological analysis.

4.3 POS Tagger

The implemented POS tagger exploits a hybrid approach which is a combination of statistical and rule-based methods to tag Persian sentences. The proposed tagger uses a novel probabilistic morphological analysis to tag unknown words. In other words it is a bigram tagger improved by probabilistic morphological analysis (Fadaie & Shamsfard, 2009).It tags the unknown words according to their internal structure in addition to their context and external environment. Besides, it learns new words and adds them to the lexicon for further use. In the proposed system a tagged corpus is needed for calculating the probabilities of morphological rules which are used in the process of tagging. The output of the tagger can be used to learn new words and enrich a lexicon. Table 12 shows some examples of the Persian POS tagger.

Input	Stems	Morphology
دانشکده هایم (my faculties)	دان Dan (know)	دان + ش + کده + هایم present stem + sh + kade + hayam
	دانش Danesh (knowledge)	دانش + کده + هایم noun + kade + hayam
	دانشکده daneshkade (faculty)	دانشکده + هایم noun + hayam
می خواند (He is reading/He was reading)	خواند khand (read past tense)	زمان: ماضی استمراری tense: past progressive نوع: سوم شخص مفرد type: third person singular می + خواند mi + past root
	خوان khan (read peresent tense)	زمان: حال اخباری tense: present progressive نوع: سوم شخص مفرد type: third person singular می + خوان + د mi + present root + third person identifier

Table 11: An example of inputs and outputs

Input	POS tag
این “in” ‘these’	Adjective
ساختمان‌ها “rekhteman-ha” ‘buildings’	Noun
را “ra” ‘This is sign of object’	Preposition
ما “ma” ‘we’	Pronoun
نساخته‌ایم “nasakhteh-im” ‘we have not build them’	Verb, Adjective

Table 12: Tagging results of Persian POS tagger

5. Experimental results

To test the different parts of STeP-1 we use a lexicon (Zaya) (Eslami, et al., 2004) containing about 57,000 Entries with their POS tags. In the first series of testing the tokenizer an input text with 400 words involving 115 errors was given to tokenizer. The performance was about 87%. The main source of errors was in segmenting unknown words to find their parts. In another test the tokenizer was applied on a text which contained 100 sentences. This text has 80 derivative affixes, 61 verb prefixes, 15 verb postfixes, 10 acronyms and abbreviations, 34 dates and numbers and 16 concatenated words. The performance of the system was 86.6%. The main sources of error has been long distance dependencies between parts of compound verbs, the lack of lexicon and considered heuristics.

The morphological analyser, with 600 input words, we got about 98% recall and 93% precision. Over-stemming is the main source of errors in this part. In this program, we cover all kinds of verbs in Persian and all the inflectional forms of words. Also we cover 45 derivational affixes. The prefix FSA has 68 nodes that 43 are finals. The postfix FSA has 184 nodes that 126 nodes are finals. Morphology contains 38 rules.

The POS tagger was tested by a 10-fold cross validation method on 2,700,000 tokens of Bijankhan corpus containing 12% unknown words. The overall precision of this tagger is 90.9% which is 11% better than the baseline (simple bigram tagger).

6. Conclusion

In this paper, we presented STeP-1, the first Persian standard text preparation system. STeP-1 receives an input Persian text and converts it into series of corrected standard tokens and their morphological stems and POS tags. STeP-1 can be used both as a style corrector and as the preprocessor in many NLP applications. The output of STeP-1 can be used in complex NLP systems for Persian language with good performance.

7. References

APLL, (2006). Academy of Persian language and literate, *Persian writing style*, Asar publication , Iran.

Brants, T. (2000). TnT-a statistical part-of-speech tagger. In *Proceeding of the sixth conference on applied natural language processing ANLP-2000*. Seattle, WA.

Dehdari, J., Lonsdale, D. (2008).A link grammar parser for Persian. In Simin Karimi,Vida Samiian, and Don Stilo, editors, *Aspects of Iranian Linguistics*, volume 1. Cambridge Scholars Press.

Eslami M., Sharifi, M., Alizadeh, S., Zandi, T. (2004). Persian ZAYA Lexicon, *1st Workshop on Persian Language and Computer*, Tehran, Iran.

Fadaie, H., Shamsfard, M. (2009). Persian POS Tagging Using Probabilistic Morphological Analysis, to appear at International Journal of Computer Applications in Technology (IJCAT), Special Issue on: *"Intelligent Text Processing with its Applications and Computational Linguistics"*.

Kiani, S., Shamsfard, M., (2008). Word and Phrase Boundary detection in Persian Texts, *14th CSI Computer Conf.*, Tehran, Iran.

Megerdoomian, K. (2004). “Finite-state morphological analysis of Persian”. *In Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, University of Geneva.

Megerdoomian, K., Zajac, R. (2000). “Tokenization in the Shiraz project”, technical report, NMSU, CRL, *Memoranda in Computer and Cognitive Science*.

Shamsfard, M., Kiani, S., Shahedi, Y. (2009). STeP-1: standard text preparation for Persian language. *CAASL-3 – Third Workshop on Computational Approaches to Arabic Script-based Languages* [at] *MT Summit XII*, Ottawa, Ontario, Canada.