STeP-1: Standard Text Preparation for Persian Language

Mehrnoush Shamsfard
NLP Research Lab.
Faculty of Electrical and Computer Engineering
Shahid Beheshti University, Tehran, Iran
m-shams@sbu.ac.ir

Soheila Kiani
NLP Research Lab.
Faculty of Electrical and Computer Engineering
Shahid Beheshti University, Tehran, Iran
So.kiani@mail.sbu.ac.ir

Yaser Shahedi
NLP Research Lab.
Faculty of Electrical and Computer Engineering
Shahid Beheshti University, Tehran, Iran
yshahedi@gmail.com

Abstract

Many NLP applications need a pre-processing task to convert the input into an appropriate form or format. The preprocessing may include segmentation of text into sentences, words and phrases, checking and correcting the spellings, doing lexical and morphological analysis and so on. The output of this phase should be a list of correct standard tokens with unique coding, spelling and prescription.

In this paper we introduce a Persian text preprocessor called STeP-1. STeP-1 performs a combination of tokenization, spell checking and morphological analysis. It 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 very good performance.

1 Introduction

Many NLP applications need a pre-processing task to convert the input into an appropriate form or format. The preprocessing may include segmentation of text into sentences, words and phrases, checking and correcting the spellings, doing lexical and morphological analysis and assigning appropriate features to the components e.g. by looking up in a lexicon. The output of this phase should be a list of correct tokens in a standard format with unique coding, spelling and writing prescription.

There are many factors which affect the complexity of the preprocessing module. For instance, languages with more different writing styles need extra preprocessing to turn all to a standard one.

Persian is among the languages with complex preprocessing tasks. One of the complexity sources is handling different writing scripts (prescriptions). In Persian there are one to four written forms for each character according to its location in a word. Also there are various scripts for writing Persian texts, differing in the style of writing words, using or elimination of spaces within or between words, using various forms of characters and so on. The interesting point is that we may write many words in more than one way. So comparing words to dictionary entries will be effective only if either we have all writing forms of all words in the dictionary or perform a preprocessing task which converts all the text to a standard writing style. These all make Persian texts hard to be processed and tokenized (Kiani and Shamsfard, 2008).

This paper introduces a Persian text preprocessor called STeP-1. STeP-1 is the first step in processing Persian language written texts. It performs a combination of tokenization, spell checking and morphological analysis.

In this paper after reviewing the related works in this field, we will describe the problems and challenges of Persian text preprocessing and then introduce STeP-1 as a solution. We will discuss spell checker and tokenizer as the main parts of STeP-1 in more details and show the experimental results.

2 Related Works

STeP-1 integrates tokenizer, morphological analyzer and spell checker and interleaves their performance. Although there is no other integration of these tasks which converts the final result into the APLL standard, we can refer to individual works on each part as related works.

In word and phrase segmentation, there are various approaches including rule based, statistical, dictionary based and learning methods.

Rule based methods need linguistic knowledge consisting of both semantic and syntactic elements. The rules may be defined by human or extracted from linguistic resources such as tagged corpora using a learning procedure.

On the other hand, Statistical approaches (such as Sanders and Taylor, 1995) do not need linguistic knowledge and their success highly depends on the resources. They are more portable, almost language independent and have shallow but wide coverage. These methods should extract statistical information such as highly frequent phrases, their frequency of occurrence and co-occurrence probabilities from processed corpus, web documents, search engine outputs and etc.

Dictionary based methods (such as X. Wang , et al. 2007) segment sentences by matching entries in a dictionary. Its accuracy is determined by the coverage of the dictionary, and drops sharply as new words appear. Use of stemming tools and morphological analysis is needed for decreasing the number of mismatches or unmatched words.

In learning methods, systems learn required segmentation information from input sources. This information can be linguistic models, semantic and syntactic rules or statistical information. In other words learning methods may be combined with each of the above approaches. Learning resources are generally lexicons and corpora. Segmented syntactically tagged corpora are one of the most appropriate linguistic resources for segmentation learning. Such corpora are not available for many languages like Persian. However, these methods properly handle new cases, but the lack of appropriate tagged corpora makes their usage difficult.

There are also different methods presented for spell checking in different languages. As simple traditional methods for spell checking are not fast and efficient, complex methods are being presented for this purpose. In this paper the idea of using web as a corpus for spell checking has been got from (Jacquemont, et al., 2007). This idea is modified and combined with classic methods to increase efficiency.

In the next section we will describe some of the problems and challenges for pre-processing Persian texts.

3 Problems and challenges

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 'پائیز' and 'پاییز' are forms of writing the word 'fall', 'مسئله', 'مسأله' and 'مساله' are all forms of writing the word 'problem' and 'حتما' and 'حتماً' are forms of writing 'certainly'.

Unicode ambiguity: there are some letters such as 'ی' (i) and 'ک' (k) for which we have two Unicode (one for Persian and one for Arabic). As some applications use the first and some the second one we have to unify their occurrences before processing the text.

Different spellings: some words may be written with different letters such as 'بلیت' and 'بلیط' for 'ticket'.

Different spacings: 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 –especially when the last character of the first word has just one form. 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', 'می‌رفت' are all forms of 'was going'.

Different writing prescriptions: APLL 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. The problem of this construction in tokenizer occurs when we write it. Ezafe will be written after long vowels such as ا، ه، و. In these places it changes to an enclitic (ی). Although these explicit enclitics may facilitate the Ezafeh detection and consequently eases chunking but they have again some different forms of writing which need some processing for recognition. For example (khane –e Ali) (Ali's House) may be written as 'خانه ی علی', 'خانه‌ی علی', 'خانه علی' or 'خانة علی' while pronouncing the same.

4 Solution: STeP-1

STeP-1 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 spacings
      i. replace white spaces with short spaces
      ii. Add white spaces (unknown words)

In STeP-1 at first the tokenizer identifies some initial tokens then it calls the spell checker to correct misspelled words and then again tokenizer moves through the text to extract corrected remained tokens.

4.1 Tokenization

Text segmentation, one of the primary activities in natural language processing; is the process of recognizing boundaries of text constituents, such as paragraphs, sentences, phrases and words.

Word Segmentation also known as tokenization focuses on recognizing word boundaries exploiting orthographic word boundary delimiters, punctuation marks, written forms of alphabet and affixes. 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; concatenates the separated parts of a single word and separates individual words from each other. It also recognizes multi part verbs, numbers, dates, abbreviations and some proper nouns.

In this section we will first discuss some of the linguistic information to be used for word segmentation and then will describe our approach.

4.1.1 Orthographic words boundary delimiters

In word segmentation, we should notice the purpose of determining word boundaries. For example, the English sentence "I'm going to show up at the ACL" has eight orthographic words separated by seven spaces. The sense of "orthographic word" is a meaningful sequence of characters. This sequence is usually written between two word boundary delimiters. If the mentioned sentence is processed for syntactic analysis, it would be probably wanted to consider "I'm" to consist of two syntactic words, namely I and am. For translation purpose, it would be probably wanted to consider "show up" as a single dictionary word since its semantic interpretation is not trivially derivable from the meanings of "show" and "up". And in text to speech synthesis, it would be probably considered the single orthographic word ACL to consist of three phonological words that are written in acronym forms (Sproat, et al., 1996). So, it can be seen that orthographic word boundaries are thus only a starting point for further analysis and can only be regarded as a useful hint at the desired division of the sentence into words.

As we mentioned above, spaces are orthographic word boundary delimiters in English. Most languages that use Roman, Greek, Cyrillic, Armenian, or Semitic scripts, and many that use Indian-derived scripts, mark orthographic word boundaries (Sproat, et al., 1996); so tokenization has less problems there. Asian languages like languages written in a Chinese-derived writing system including Chinese, Japanese, Korean and Vietnamese do not delimit orthographic words. These languages are hard in determining word boundaries, as well as have different phonetic, grammatical and semantic features from Indo-European languages (Nguyen, et al. 2006). Although Thai has Indian-derived writing systems but unlike English has no word boundary delimiter (Tesprasit, et al., 2003).

Word boundary determining and automatic orthographic words detection have also some problems and complexities in Persian. Although Persian is an Indo-European language and words are usually separated by spaces but because of various prescribed forms of writing in this language, spaces are not the exact and deterministic boundaries of distinct words. In written Persian texts, we have two kinds of spaces (APLL,2006):

- In word space: spacing is used between components of compound nouns usually in detached characters. This space is known as short space.
- Between word space: spacing is used between distinct words of a phrase or a sentence. This space is known as white space or space.

In other words in Persian, distinct words should be separated by white spaces and components of derived and compound nouns and inflectional structure of verbs should be delimited by short spaces. So, we will have problems with situations in which there should be no space but spaces exist and situations in which there should be spaces but no space exists. As an example, consider word "میرفتم" (miraftam) (I was going). It can be written in 3 correct ways: "میرفتم" (miraftam) , "می¬¬رفتم" (mi~raftam) and "می رفتم" (mi raftam).

These forms are all correct in various prescribed forms of writing and automatic word segmentation systems should detect all forms and produce a single unique form of them for further analysis.

4.1.2 Punctuation marks

Punctuation marks determine the end of phrases and sentences and can help text segmentation. Punctuation marks play an important role in text to speech synthesis; because these marks make differences in pronunciation of phrase and sentences. For example a sentence ends with a dot or exclamation, will be pronounced in different forms. In many languages like English, punctuation marks exist but in some languages like Vietnamese do not (Tesprasit et al., 2003). In Persian we have punctuation marks and can use them in segmentations (Megerdoomian , Remi Zajac, 2000).

4.1.3 Written forms of characters

In English and similar languages that use Latin characters for writing, characters have two forms: small and capital. In most cases, small letters are used for writing and capital letters are used less just in abbreviation, acronym and initial letter of proper nouns, so written forms of characters can not be used for segmentation in these languages.

The Persian writing system distinguishes between the final, initial and medial forms of a character, depending on its position within a word. Some characters only have a single form that does not depend on their position in a word. In these cases we can not use final forms of characters for words boundaries detection

An initial form does not mean that the character is in the beginning of a word, it only indicates that the character is not at the end of the word. Characters are in medial form if they are attached from both sides. A final form character indicates the end of a word (or subword) and can be used by the tokenization system to determine word boundaries. Hence, two concatenated words can be put into separate tokens if the first word ends in a final form letter (Megerdoomian , Remi Zajac, 2000).

4.1.4 The Proposed tokenizer

Our proposed system combines dictionary based and rule based approaches. We use punctuation marks, spaces and morphology analysis to segment words. Figure 1 shows the overall architecture of the system. It contains the following steps:

- **Split input text due to white spaces**
- At first, we segment input text according to white spaces.
- **Number Processing**

Numbers should be written with white spaces same as words in the text. However, white spaces may be deleted without making any ambiguity. The same fact is true about dates and floating point numbers which are written by slash.

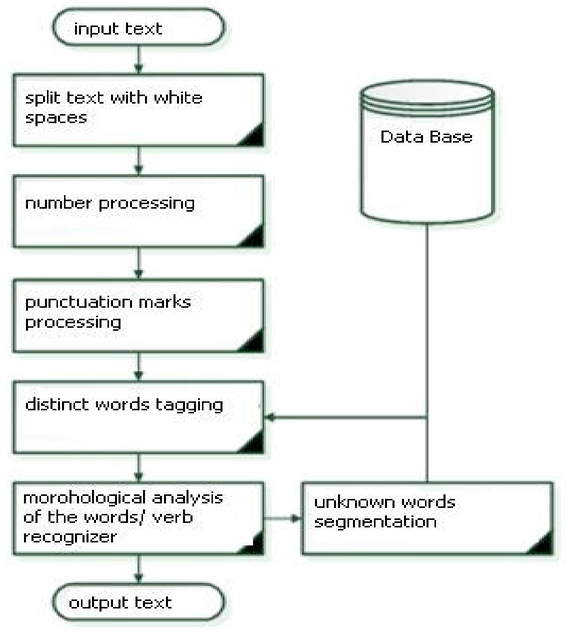

input text
split text with white spaces
number processing
punctuation marks processing
distinct words tagging
morohological analysis of the words/ verb recognizer
unknown words segmentation
Data Base
output text

Figure 1. the proposed tokenizer

- **Punctuation marks processing**

Input text usually has many punctuation marks that have different positions across words. The entire position pattern is:

Word | punctuation mark | white space | word

But written text may have less/more spaces, for example a space is inserted between first word and punctuation mark or white space is omitted.

At the first step we amputate all punctuation marks from text, but we save their position and their kind to insert them correctly at the end. Punctuation marks also help us to adjust prefixes and postfixes.

- **Distinct words tagging**

By use of a dictionary of nouns, we can tag some few segments as words to reduce computational complexity, because the most of words are in their inflectional or derivational form (e.g. plural forms or having affixes).

- **Morphological analysis of nouns**

Segments which are separated by white spaces and have not been tagged as a noun from dictionary would be checked from the end according to the following pattern:

Derivational prefixes | noun | derivational postfixes | comparative/superlative morpheme | plural suffix | possessive pronouns/enclitic morpheme

Containing all these parts mentioned in the above pattern is not obligatory. For unification reasons, if the plural suffix “ha” attached to word, it will be detached and will be rewritten by adding a short space during this check. The aim of this phase is to find out whether a segment is a word with some affixes or just a group of affixes without any distinct word. As the number of affixes is limited compared to words, in this phase we try to separate and recognize the affixes rather than stems. If it is not a derivational affix, it will be considered as a word. This will lead to recognition of unknown words as noun accurately.

In Persian, some derivational affixes are words themselves and can be used individually. In order to recognize these affixes correctly, semantic analysis and context processing are required.

In this research we consider such affixes as nouns. For example consider the word “خوشحال“ (khoshhal) (happy), it may be written as “خوشحال“ or “خوش¬حال“ or “خوش حال“. In the last case in which the two parts of the word are separated, we have a special situation in which both parts can be a separate meaningful word. In these cases we can not reduce the space by a short space unless doing some context analysis. So we keep the current writing unchanged.

One of the advantages of this approach is that by doing some morphological processes in this step, we extract the stems which are very useful for further processing of the text.

- **Verb recognizer**

In Persian some verbs have prefixes which should be concatenated to the verb by a short space. To recognize these cases we use a table containing valid prefixes for some verb stems.

In some cases we may have different spacing rules for different number and person of verbs. For example in past participles the postfix of single third person should be separated from the stem by a space while in other number/persons of the same verb the postfix should be separated by a short space. To reduce the needed memory size, we just keep the present and past stems of verbs.

In Persian, past continuous and present continuous verbs consist of a prefix “mi”. This prefix has a homograph which is a separate word and means wine. To recognize this (if it is not tagged in previous steps), we should check the next word. If it is a verb then we should standardize the various writing styles of the verb by inserting a single short space between the prefix and the stem. A similar case occurs for future tenses of verbs too.

- **Add white spaces to determine unknown words**

As we mentioned before, some Persian characters have a single form and so the words ending with them may be written with no space besides the next word. It can be seen in concatenated words too which are very common in Persian text. This form of writing is not incorrect but should be transformed to the standard form by inserting spaces. Determining these cases is not easy. To do this, we should find all permutations of the sub-words ending by a single-formed character in a sequence. It can be done by inserting a space following characters without a final form. These characters are alef(a), dal(d), zal(z), re(r), ze(z), zhe(j), vav(v). Since one or both words may be inflected, the separated tokens need to also undergo morphological analysis. For all occurrences of the above characters the tokenizer produces two segmentations: one in which a space is inserted following these characters and one without a space. A space need not be inserted after the final character in the string. We could eliminate some of the unwanted cases by discarding any combination that contains a single letter (except for "v", which is the conjunction 'and'). It is also possible to eliminate certain combinations if the last string is a suffix. For instance, if the string "مسافرين" (mosaferyn) (travelers) results in the segmentation "mosafer" "yn", since "yn" is a plural suffix not a known word, this particular segmentation will be eliminated. Then the sub-words should be checked to find if they are a known word or a derivation of a known word or not. The output of this phase is all alternative cases with various spacing which have meaningful constituents (Megerdoomian , Remi Zajac, 2000).

4.2 Spell checking

There are some spell checkers for Persian language. Although some of these methods are efficient, all of them have a common problem. The problem is that in these methods the spell of each word is checked separately. In other words, the position of the word in the sentence and its context are not mentioned during the spell checking.

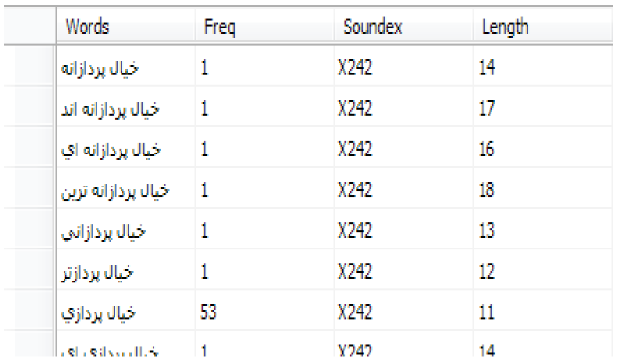
Two different solutions could be used to solve the problem. First one is to check grammatical rules in sentences and the second one is to use statistics of using different phrases in a language. According to a large number of exceptions in Persian grammar and also some complexities of this language, it is so difficult and time consuming to check grammatical rules for spell checking in Persian texts. Therefore, the second method is much more efficient.

In this paper we present a novel method for spell checking in Persian (Farsi) language. We use web as a corpus in our method to modify previous classic offline methods. Using this online corpus along with an ordinary offline database makes the method much stronger and accurate. Results show that the presented method is able to detect different kinds of misspelling errors more efficient than previous spell checking methods.

In this section, we will first present a basic method of spell checking in Persian based on a database which is generated from Hamshahri corpus[1]. Then we describe our statistical method which uses web as a corpus.

The Basic Spell Checker

Classically, Spell Checkers use a dictionary as their database. Here, we generate and use our own database of Persian words and their features. To do so, using our tokenizer, we extracted a database from Hamshahri corpus containing about 900000 records of words, their soundex code, their frequency of occurrences, and their length. Figure 2 shows a part of this database.



Words	Freq	Soundex	Length
خيال پردازانه	1	X242	14
خيال پردازانه اند	1	X242	17
خيال پردازانه اي	1	X242	16
خيال پردازانه ترين	1	X242	18
خيال پردازاني	1	X242	13
خيال پردازتر	1	X242	12
خيال پردازي	53	X242	11
خيال پردازي اي	1	X242	14

Figure 2. a part of database

By using the adapted soundex code for Persian we can put all words which are different but are similar in sound, a common phenomenon in Persian language, in one class.

[1] http://ece.ut.ac.ir/DBRG/Hamshahri/fa.htm

Candidate Suggesting Spell Checker

This spell checker is used to extract candidate suggestions for both correct and incorrect words. Then a web-base algorithm which is presented in the next section uses these suggestions to decide. If we consider a standard spell checker, its basic principle is quite simple. Let T1 ... Tn be the constitutive words of a document. When the spell checker detects Ti that is not in its associated dictionary, it searches in this dictionary for some candidate suggestions that could be suggested to the user in order to correct this misspelling. Here we do the same thing also for Ti that is in the database and extract the appropriate candidate suggestions. To obtain candidate suggestions, all types of errors should be checked. In missing a letter, miss double letter and wrong letter errors, first we search in database for words whose frequency is more than 100 and their difference with spell checking word is in one letter, and also their soundex codes are the same. Then the distance between them is measured. The word or words with distance of one is selected as candidate suggestions. For wrong located letter error, words with the distance of two and the same length and also the same summation of their letter ASCII codes are selected. For miss merge error, the word is divided to two separate parts and each part is searched in the database. If both parts are in the database, the miss merge error is possibly occurred and these two words are mentioned as candidate suggestions too. The extracted words from database are used for next part of the algorithm which is the Web-based spell checker query.

Using Web as a Corpus

After extracting appropriate candidate suggestion words for each word of the text, we search for errors and their corrections on the web. Any search engine can be used for achieving this goal. Here we use Google search engine.

For searching phrases in Google we design a web service that receive a query string as an input that contains many phrases, then the web service searches every phrase in Google and make a query string which contains the phrases and the number of search results and send it as its output

For example, a web service input would be as follows:

L0=وب سرو سي چيست&L1=وب روسي چيست&L2=وب دروسي چيست&L3=وب سروري چيست&L4=وب سرويس چيست&L5=وب سيروس چيست&L6=وب عروسي چيست
(L0=web serv si chist&L1=web roosi chist&L2=web droosi chist&L3=web sarvari chist&L4=web service chist&L5=web siroos chist&L6=web aroosi chist)

And also its output would be as follow:

وب سرو سي چيست=0&وب روسي چيست=0&وب دروسي چيست=0&وب سروري چيست=0&وب سرويس چيست=2360&وب سيروس چيست=0&وب عروسي چيست=0
(web serv si chist=0&web roosi chist=0&web droosi chist=0&web sarvari chist=0&web service chist=2360&web siroos chist=0&web aroosi chist=0)

In this section the algorithm has two main parts, one apply for incorrect words and the other for correct words. For both incorrect and correct words, every candidate suggestion for the misspelled word (CS) is searched on the web with its previous (Tp) and next (Tn) words and the number of search results is used for obtaining a criterion of checking errors. For correct words, the word itself is also searched. The following logarithmic formula generates the criterion for compare the search results:

$$P = 1 + \frac{1}{9}\log\left(\frac{\#occ(T_p.Cs.T_n) + 1}{10^9}\right)$$

where #Occ(Tp.CS.Tn) is the results number for searched tri-gram phrase. This formula will return a value between zero and one. A bigger value means a better word to choose. The obtained results for different candidate suggestion are compared and the best word will be chosen as the best candidate for correction.

One of the cases in which the algorithm may not work well is a continuous string of misspelled words in a sentence. To solve this problem, every candidate suggestion for the misspelled word is searched on the web with its previous or next word which is correct, but if both words are misspelled the algorithm searches misspelled word with previous word which is corrected in previous step.

Comparison to the method presented by Jacquemont and colleagues (2007) shows that the number of searches in our method is smaller and therefore it is faster.

Although the results for such an algorithm can be much accurate, by searching different styles of word phrases, increasing the number of searches will increase the running time. Hence, we use the tri-gram method to optimize the algorithm.

5 Experimental results

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 Proposed Spell Checker is applied to three different texts with 100 misspelling errors in each one. Each text contains about 500 to 600 words. The texts are selected from three different categories; novel, history and general texts. The texts contain misspelling errors of 4 different types. Thus, there were 300 different errors in our test texts. Following in table 1, the practical results of applying our algorithm to these texts are presented. In this table 'wrong' means words with a wrong letter (need substitution), 'missed' means Words with a missed (deleted) letter (need addition), 'extra' means words with a wrong extra letter (need deletion) and 'merged' means errors in which two separate words were merged by removing the space between them (need space insertion).

Test	type	Error #	Detection #	detection rate	Correction #	correction rate
1	wrong	25	25	100	24	96
	missed	25	24	96	24	96
	extra	25	25	100	23	92
	Merged	25	25	100	22	88
	All	100	99	99	86	86
2	wrong	25	25	100	25	100
	missed	25	25	100	24	96
	extra	25	25	100	25	100
	Merged	25	25	100	23	92
	All	100	100	100	97	97
3	wrong	25	25	100	23	92
	missed	25	25	100	24	96
	extra	25	25	100	25	100
	Merged	25	25	100	22	88
	All	100	100	100	95	95

Table 1:Results of algorithm on three different texts

Overall, our algorithm could detect about 99.6% of errors and correct about 92.6% of them.

As an example an input sentence (errors are underlined) and its corresponding corrected sentence is shown here:

Input:

در نگاه نحست به نذر مي امد که آموختنافعال بي قاعده(فارسی
)۱۰۰۰باردشوار تر از آموختن افعال با قاعدهاست

Output:

در نگاه نخست به نظر می‌آمد که آموختن افعال بی‌قاعده (فارسی)
۱۰۰۰ بار دشوارتر از آموختن افعال باقاعده است

6 Conclusion

In this paper, we have 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 inflectional stems. It performs spell checking and morphological analysis within a tokenizer. STeP-1 can be used both as a spell and style corrector and as the preprocessor in many NLP applications.

References

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

S. Jacquemont, F. O. Jacquenet, M. Sebban. 2007, *Correct your text with Google*. IEEE Int'l Conf. on Web Intelligence, Silicon Valley, USA.

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

K. Megerdoomian , Remi Zajac, 2000, *Tokenization in the Shiraz project* , technical report , NMSU, CRL, Memoranda in Computer and Cognitive Science.

T. V. Nguyen, H. K. Tran, T.T.T. Nguyen, Hung Nguyen, 2006, *word segmentation for Vietnamese text categorization: an online corpus approach* The 4th IEEE Int'l Conf. in Computer Science, Hochimineh, Vietnam,

E. Sanders , Paul Taylor , 1995, *using statistical models to predict phrase boundaries for speech synthesis* , EUROSPEECH '95 , Madrid, Spain

R. Sproat, C. Shih , W. Gale , N. Chang ,1996, A stochastic finite-State word-segmentation algorithm for Chinese , *Computational Linguistics* , Volume 22 , Issue 3, Pages: 377 – 404

V. Tesprasit , Paisarn Charoenpornsawat and Virach Sornlertlamvanich *, learning phrase break detection in Thai text-to-speech* ,2003, '8th European Conf. on Speech Communication and Technology, Geneva, Switzerland.

X. Wang , W. Liu , Y. Qin , 2007, *a search-based Chinese word segmentation method* , 16th Int'l Conf. on World Wide Web. Banff, Canada.