FarsiSpell: A spell-checking system for Persian using a large monolingual corpus

Tayebeh Mosavi Miangah
Payame Noor University, Islamic Republic of Iran

Abstract

In recent years, great availability of various language resources in different forms as well as rapid development of computer technology and programming skills have made researchers in the fields of linguistics and computer science cooperate in solving different problems of computational linguistics and natural language processing. Building large monolingual as well as bilingual corpora in digital forms and storing them in computer memories has enabled linguists and language engineers to automatically explore techniques for processing information with the help of various computer programs without any need to manually collect and analyze data.
One of the main applications of monolingual corpora can be seen in developing automatic spell-checking systems. In such systems, a large monolingual corpus can function as a database instead of a monolingual dictionary. In the present study, it has been tried to demonstrate the effectiveness of a large monolingual corpus of Persian in improving the output quality of a spell-checker developed for this language.
In the present spelling correction system, the three phases of error detection, making suggestions, and ranking suggestions are performed in three separate stages. An experiment was carried out to evaluate the performance of the spell-checking system.

Correspondence:
Tayebeh Mosavi Miangah, Linguistics Department, Payame Noor University, Iran, Islamic Republic of Iran.
Email:
Mosavit@pnu.ac.ir or Mosavit@hotmail.com

1 Introduction

We live in the world of technology where the huge available information should be processed and translated into knowledge to be accessible and applicable for further applications. However, traditional methods of information gathering and storing will no longer be sufficient in the coming centuries. Exploiting modern technology in solving various problems of language studies can be regarded as an efficient way in learning environments.

Corpus-based linguistics has provided an accurate description of language, and its new potentials for language structure and use have many applications in linguistics and some other related fields.

Recent availability of large monolingual as well as bilingual corpora in digital and online forms enables linguists and language engineers to automatically explore techniques for processing information directly from different types of linguistic texts. It provides them with great opportunities in language analysis, quickly and accurately, with the help of various computer programs without any need to manually collect and analyze data.

Spell-checking can be considered as one of the main applications of large monolingual corpora. Developing human–computer technologies has emerged some novel applications demanding misspelled words identification and their correction abilities. The majority of spell-checkers need to have a dictionary to be used as a database in which all and only legitimate words of the given language occur. It is obvious that no dictionary includes all different paradigms of a given word or most of proper nouns, specifically person names. So, it is necessary to incorporate a lemmatizer and/or morphological analyzer as well as a table of all existing proper nouns of the language to the database of the system, otherwise the output would be insufficient. One practical solution to this problem is to have a sufficiently large monolingual corpus to be used instead of a dictionary as a database in a spell-checking system. The general idea of the present article is to focus on this very issue.

To make the readers well-informed with the whole process of our spell-checking system, Figure 1 can be of great help in this respect:

The process of spell-checking, firstly, is divided into three phases: Detection of errors or misspelled words, during which the lexical analyzer detects the misspelled word in the input string. Making suggestions for detected misspelled words, during which the system creates a set of possible candidates as potential replacement for the given erroneous word. And ranking as well as automatic correction of errors, during which these candidates are sorted out from the most likely replacements to the least likely ones based on their associated error weight. Most techniques treat each phase as a separate process and perform them in sequence.

The following parts of this section are the attempts to give a detailed description of these three phases, different types of errors, and sources of errors in general as well as in Persian language. Section 2 tries to provide readers with some information about major properties of Persian language, especially morphological patterns in this language. In section 3, a review of some recent works on different approaches toward spell-checking is provided. Section 4 deals with describing methodology applied in the experiment carried out in this article. Section 5 thoroughly describes the processes involved in performing the experiment using our large monolingual corpus of Persian. Sections 6 and 7 are dedicated to analyzing the results from the experiment, and conclusion and further developments, respectively.

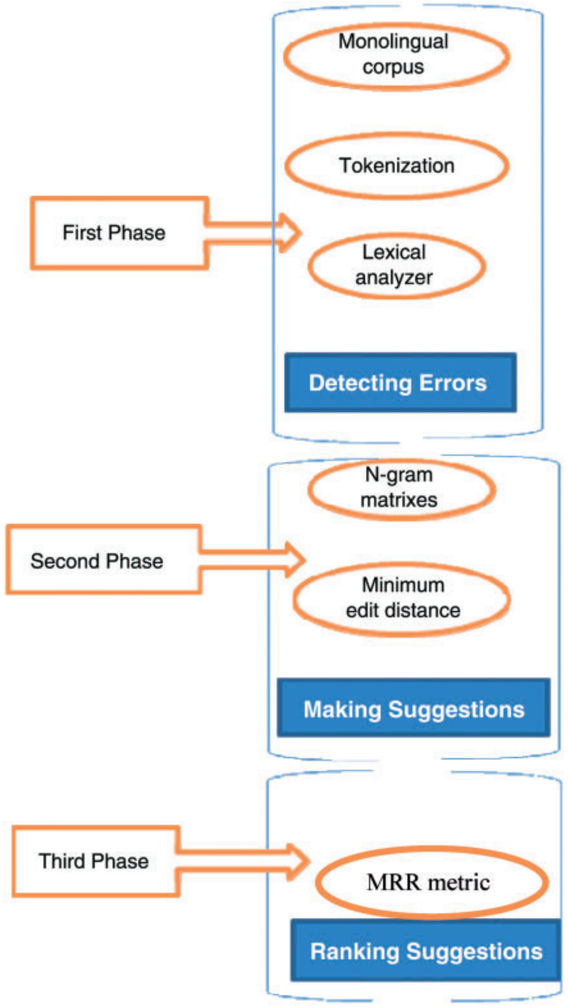


Fig. 1 Spelling-checker algorithm flowchart.

1.1 Detection of errors

The first step in spell-checking process is correct detection of an error or a misspelled word. This process usually consists of looking for each input word in a dictionary or database to find out if it is a valid word belonging to that dictionary or database. Here there are two possibilities, as follows:

(a) The detected misspelled word is a valid word. That is, the word detected as an error is among the existing words in database of the system. They are usually referred to as 'real word errors'. An example for this case is the word 'after' instead of 'alter', or امت instead of املت.

(b) The detected misspelled word is a non-word. That is, the word detected as an error cannot be found in database of the system. Replacing 'tgis' instead of 'this', or سخنی instead of سحنی, may be an example for this case.

Detecting the errors of the first type is not in the scope of the present investigation, as in most cases, it requires context analysis and a robust intelligent syntactic parser and a reliable stemmer for the given language. However, some researchers tried to deal with real-word errors with the help of statistical methods. To mention a case, Faili used Mutual Information as well as a confusion set for every Persian word. Based on his report, his method got the 79% performance both in detecting the error and non-error correctly (Faili, 2010). In a similar study, Faili and his colleague presented a language-independent approach based on a statistical machine translation framework to deal with context-sensitive spelling errors or real-word errors and grammatical errors in Persian language. They reported encouraging results out of their experiments (Ehsan and Faili, 2011), and we confine our study to the second case. In its simplest form, a spell-correction system reads the input string word-by-word, searching each one in its own database. If the given word is not available in the database, it will be marked as a misspelled word or an error. The proper nouns, some inflections, as well as out-of-vocabulary words also follow the same rule. That is, they are considered misspelled words, unless the identical cases be found in the database. This way, the error is detected by the system subjected to the subsequent steps to be corrected.

1.2 Candidate generation for errors

A high-quality spell-checker should have the capacity not only to detect misspelled words, but also to provide users with some possible suggestions for every already detected error, as the users expect spelling checkers to suggest corrections for non-words.

There are a variety of approaches toward arranging possible suggestions for each error detected, such as minimum edit distance techniques (Damerau, 1964), similarity key techniques (Zobel and Dart (1996), rule-based techniques (Yannakoudakis and Fawthrop, 1983b), n-gram-based techniques (Ullman, 1977), probabilistic techniques (Pollock and Zamora, 1984), dynamic programming techniques (Oommen and Kashyap, 1998), and neural nets (Hodge and Austin, 2003). However, the most common techniques for organizing a set of possible suggestions use a dictionary or a database of legal n-grams to locate some possible correction terms (Golding and Schabes, 1996).

1.3 Ranking and automatic correction of errors

After detecting erroneous words and providing one or more potential suggestions, it is time to sort out these candidates from the most likely replacements to the least likely ones. Using various techniques, an error weight is associated with each candidate. Then, the list of suggestions is sorted in descending order. Having this, automatic correction usually takes the first candidate as the correct replacement for misspelled word. In interactive systems in which there is a kind of interaction between user and computer, ranking is the task performed by the computer, whereas selection is left to be done by the user.

The most usual techniques for ranking process use some lexical-similarity measure between the misspelled word and each potential candidate or a probabilistic estimate of the likelihood of the correction to rank order the candidates (Kukich, 1992).

At present, the system presented in this study has not the potential for automatic correction of errors. The possible suggestions are displayed according to their degree of edit distance, from the minimum edit distance to the maximum one. In fact, it is a kind of automatic ranking, from the most likely occurring suggestions to the least likely ones. The selection of a possible substitute for the erroneous word will be left to the user.

1.4 Types of errors

As the previous researches in spell-checking revealed, there are four types of errors responsible for making a vast majority of misspelled words in any language (Damerau, 1964). These four error types are as follows:

(a) **Insertion:** adding one or more characters in front, at the end, or between any two characters of a valid word to make it a misspelled one, examples:

assuyme instead of *assume*
اتاقف هایی instead of اتاق هایی

(b) **Deletion:** deleting one or more characters from a valid word, examples:

corection instead of *correction*
بیدری instead of بیداری

(c) **Substitution:** changing a character into another character, examples:

pussible instead of *possible*
حنگامه instead of هنگامه

(d) **Transposition:** switching a character with another one (usually the one next to it), examples:

recieve instead of *receive*
فلاکس instead of فلاسک

The aforementioned error types when involved in one character are often referred to as single-error misspellings, as opposed to dubbed multi-error misspellings, the errors involved in more than one character to be deleted, inserted, substituted, or transposed. The latter case is true for errors caused by two error types together. For instance, it is possible for the writer to type 'moultipel' instead of 'multiple', to which the two processes of insertion and transposition have been applied. It is also possible that a valid word of a given language is produced while a kind of mistyping occurred. For example, the word 'ate' may be produced when the intention is 'are', or the Persian word 'مسکن' (house) may be replaced by the word 'مشکن' (don't break) or 'ممکن' (possible). Correcting these types of errors that lead to a valid word needs a spell-checking system that uses context and/or is provided with a syntactic analyzer (although shallow one).

Also, it should be noted that the types of errors committed by an OCR[1] program may be different from those committed by a human being. The majority of OCR-generated errors are of the type substitution. Studies show that the error types related to OCR are various depending on factors such as font input quality, etc., ranging from recognizing one character instead of the other one ('o' instead of 'c') to taking two or more characters as one and vise versa ('iii' instead of 'm' or 'd' instead of 'cl') (Johns *et al*., 1991).

1.5 Sources of errors

There are a variety of causes for a misspelled word to be produced by a writer or a typist. As the focus of this article is going to be on tackling errors leading to non-word errors, the error sources of this type are to be taken into account. Three sources for non-word misspellings are often distinguished as: (1) typographic errors, (2) cognitive errors, and (3) phonetic errors (Kukich, 1992).

Typographic errors are those produced due to writing slips. Although the correct form of the word in question is already known, it is mistakenly misspelled by the writer. Writing 'typas' instead of 'types' or 'تخلیل' (non-word) instead of 'تحلیل' (analysis) may be good examples for the typographic errors.

Phonetic errors are divided into two types. The first one caused by the writer's lack of knowledge on how to write the correct form of the word. Replacement of 'development' by 'developement' and 'انتظار' (waiting) by 'انتضار' (non-word) are examples for the second case. In Persian, there are several letters having different forms, which give rise to serious problems for incompetent Persian writers. For instance, there are four different forms for a single character sounding ‘z’ as ‘ز، ذ، ض، ظ’ and three forms for a single character sounding ‘s’ as ‘س، ص، ث’ in Persian alphabet. Every word containing one of these various forms might be written in alternative form due to the writer’s misconception. The second one is a type of error in which an assumed phonetically correct misspelled form is produced by a really correct word. When the word ‘speshle’ is written instead of the word ‘special’ or the Persian word ‘خاهر’ (non-word) instead of ‘خواهر’ (sister), phonetic error has occurred. These types of errors are frequently seen in writing unfamiliar names and words.

In this study, it has been tried to develop a robust program used to detect and correct mistyped words of Persian. The problem is to be solved by checking whether a word already exists in the corpus (instead of dictionary). In case the word was not found, the program detects it as an error. As it is not always clear which word was intended by the writer to be replaced the misspelled word, the program provides the user with a range of suggestions to be selected as the replacement for the erroneous word. That is, it tries to extract words from the corpus that are most similar to the word in question. The list of suggestions is produced based on minimum edit distance to the misspelled word without taking context into account. Letter N-grams (bi-gram and tri-gram) are also incorporated in the minimum edit distance process to enhance its performance.

In some spelling correction systems, the three phases of error detection, making suggestions, and ranking suggestions or automatic correction are performed in three separate stages. However, there are some other systems in which these three phases are to be executed in one step. The technique presented in this study divides the spell-checking process into three phases and goes through the first two ones, leaving the third phase for a separate study. However, in the third phase, we manage to rank the candidate suggestions generated in the second phase according to their relative frequencies in the corpus. Our goal is to develop a spelling checker that can treat only non-word errors in Persian effectively and quickly.

2 Persian Background

Persian is a member of synthetic language family. It means that in Persian, a new word is to be created by adding prefix, suffix, infix or another noun, adjective, preposition, or verb to the beginning or the end of the word or verb stem. In these cases, the basic form of the word or verb stem usually is not broken. (Mosavi Miangah, 2001). Grammatical word order of Persian is subject–object–verb (SOV), although a relatively free word order is also possible, but not grammatically acceptable. In Persian, every verb has two stems, present stem and past stem, and different inflectional forms of a verb are constructed either using the present stem or the past one.

In addition to verbs, many nouns, adjectives, and adverbs in Persian are constructed from the present or past stem of the verbs. In these cases, we name such words as ‘derivative’ words, as opposed to ‘concrete’ (primary) words, in which no verb stem is involved. Nouns دانش or *da:nsh* (science) and دانشگاه or *da:nshga:h* (university), adjectives دانشمند or *da:nshmnd* (scientist) and بی دانش or *bida:nsh* (ignorant), as well as the adverb دانشمندانه or *da:nshmnda:nh* (scholarly) have been constructed from present stem دان or *da:n*, which means ‘know’ in English.

The elements within a noun phrase are linked by the enclitic particle called **ezafe**. This morpheme is usually an unwritten vowel, but it could also have an orthographic realization in certain phonological environments (Megerdoomian, 2000). For example, when the last letter of the first noun is ‘a:’ or ‘u’, we have to add an ‘-e’ to it to combine the next noun or the next adjective (notice that in Persian, the adjectives precede the nouns). Consider the noun phrase آهوی نر or *a:hu-e nr* (red deer or stag) in which ‘–e’ is the realization of ezafe between the two nouns. In the most other cases, the ezafe is pronounced as ‘e’ but it is not written. The realization of this enclitic particle in Persian texts is a challenging problem in the tokenization part in the spell-checking system of this language.

Adjectives follow the same morphological patterns as nouns. They can also appear with comparative and superlative morphemes. Certain adverbs, mainly manner adverbs, can behave like adjectives and can appear with all the adjectival affixes (Megerdoomian, 2000).

The inflectional system for the Persian verbs consists of simple forms and compound forms; the latter are forms that require an auxiliary verb. The simple forms are divided into two groups according to the stem they use in their formation, present or past. The citation form for the verb is the infinitive (Megerdoomian, 2000).

Although inflectional morphemes can attach to the verbs for conjugation, they also may attach to the end of the nouns or adjectives and convey a complete sentence enclosing subject and verb. However, in these cases, the verb can only be intransitive and in the form of ‘be’. Consider the following example:

> فقیر or *fqir* (poor) + اید or id = فقیرید or *fgirid* (you are poor.)
> کتاب *ketab* (book) + های *ha:ye* (s) + مان *ma:n* (our) = کتابهایمان (our books)

The vast variety of inflectional and derivational suffixes in Persian causes some complexities in computational detachment of Persian words, which, in turn, makes the process of detecting and correcting the errors more difficult in this language.

In Persian writing system, letters in a word are often connected to each other. Most characters have two or three alternative forms depending on their position within the word. The initial form indicates that no element is attached to the element from the right. Note that 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 have an attaching character both before and after them. The final form denotes that the character is at the end of a word. The final forms can therefore be used to mark the word boundaries. However, certain characters have only one form regardless of their position within the word.

In most languages, there is often a space between two separate words; however, in Persian, this is always not the case. There are some cases in which a space occurs within a single word. In such cases, the two parts of the mentioned word can be written totally separated, connected to each other, or using a zero-width non-joiner character (ZWNJ) to keep the two parts closer. The third possibility is the only acceptable format of writing for such words. As all Persian writers do not observe this rule, there are so many problems for tokenizing Persian texts in this connection.

Compounds and detachable morphemes (i.e. morphemes following a word ending in final form character), however, are written without a space separating them. In other words, the two parts of a compound appear next to each other but the first element in the compound will usually end in a final form character; hence it would be possible to recognize the two parts of the compound. This form is not very consistent, however, and sometimes words can appear without a space between them. If the first word ends in a character that has a final form, then we can easily distinguish the word boundary. But if the first word ends in one of the characters that have only one form, the end of the word is not clear. Although this latter case is usually avoided in written text, it is not rare. Furthermore, a space is sometimes inserted between a word and the morpheme. In such cases, the morpheme needs to be reattached (or the space eliminated) before proceeding to morphological analysis of the text (Megerdoomian, 2000). Considering word inter-spacing and intra-spacing issues in Persian language is among the key points in designing a robust spell-checker for this language.

3 Related Work

Automatic spell-checking and spell-correction systems have a relatively long story, probably as long as the computers. In fact, work on these systems began in 1960s and is continually developing up to the present. Many researchers tried to tackle the problem in distinct phases as detection, suggestion, and correction, separately. Some studies only deal with the first stage, whereas some others treat the first two stages or the three stages altogether. The Unix^{TM} spell program, for example, is an effective and efficient program for spelling error detection that deals only with the first stage. This program takes a document as input, searches each string in a dictionary consisting of 25,000 terms in the field of technical writing, and then generates a list of all strings that did not belong to the dictionary. The program does not provide any means to suggest some candidates or correct the misspelled words, leaving the task to the user (McIlroy, 1982). Grope, on the other hand, is another Unix tool developed to deal with the second phase—creating a list of candidate suggestions for the misspelled word. It is, in fact, in the form of an interactive system in which after detecting typos, the user is encountered by an ordered list of suggested words based on similarity metrics, among which s/he can choose the best one (Taylor, 1981).

Some researchers have used rule-based techniques for error detection and correction. Rule-based techniques are heuristic programs using the knowledge of the most common error types to extract some rules to be able to correct a misspelled word. Yannakoudakis and Fawthrop used the knowledge of predictable errors to convert the non-word into lexicon words. They extracted the rules from 1,377 misspelled words. The rules display various spelling errors so that they can be applied to different non-words to create corresponding correct words. They tested their program on a set of 1,534 misspelled words selected from a variety of sources. They found that a large portion of the errors could be dealt with by a set of seventeen heuristic rules, twelve of which related to the misuse of consonants or vowels in graphemes, and five of which related to sequence production (Yannakoudakis and Fawthrop, 1983a,b).

Some other studies, like that reported by Ahmed Hassan and his colleagues, tried to find appropriate solution for the problems of detecting misspelled words, generating candidate corrections for them, and ranking corrections. They used finite state automata for automatic correction of spelling mistakes to propose candidates corrections within a specified edit distance from the misspelled word. First they provided a list of some candidate suggestions for the misspelled word using minimum edit distance and then they used a language model to assign scores to these candidate suggestions and choose best correction in the given context. Their presented approach is claimed to be a language-independent one and requires only a dictionary and text data to build a language model. They tested their approach on both Arabic and English text and achieved accuracy of 89%. This accuracy deals with testing the auto-correction process using a list of 556 words having common spelling errors in both languages (Hassan *et al.*, 2008).

Fossati and Eugenio addressed the problem of real-word spell-checking, i.e. the detection and correction of errors that result in real words of the given language, such as *peace* and *piece*. Although handling these types of errors is far more intricate than non-word errors, as the former needs context information, they used a mixed trigrams language model to present a statistical method for context-sensitive spell-checking. Their model was implemented, trained, and tested with data from the Penn Treebank and showed promising results with respect to the hit rates of both detection and correction (Fossati and Eugenio, 2007).

In another study, Zamora *et al.* tried to evaluate the effectiveness of trigram frequency statistics for spell-checking applications. They compiled a trigram matrix in which instead of binary values, the frequency counts or probabilities extracted from a sufficiently large corpus of text (e.g. at least a million words) were shown. To determine the difference between the trigram compositions of correct and misspelled words and see whether the misspelled words are reliably detected, they analyzed a set of 50,000 word/misspelling pairs from seven machine-readable databases and concluded that trigram analysis can detect the error position in a misspelled word accurately, but it cannot distinguish between valid words and erroneous ones (Zamora *et al.*, 1981).

Colak and Resnik took a pattern recognition approach to string error correction of misspelled words using OCR. Their pattern-recognition approach treats OCR as within the framework of the noisy channel model, a model with attractive theoretical properties, contributing new techniques for empirical parameter estimation. Their approach was reported to work significantly better than the naive approach of correcting errors by finding the most similar dictionary entry according to simple edit distance. They evaluated their model on artificial as well as real data and observed that the size of the candidate dictionary affects correction accuracy (Kolak and Resnik, 2002).

Mitton presented a developmental study of spell-checking systems and tried to investigate different approaches toward the problem. According to Mitton, the spell-checking has a rather long history. It began in the late fifties—the days of mainframes and punched paper tape (Mitton, 2010).

As far as Persian language is involved, there have been a few research studies in building or developing spelling checkers in this language. Rezvan and his colleagues—while introducing FarsiTEX, a document preparation tool in Persian language based on LATEX—discussed some difficulties of spell-checking as well as some suggestions in this language (Rezvan *et al.*, 1996). In another study, Saburian and Dorri, discussing different components of a spell-checker, introduced a comprehensive approach toward implementation of a Persian spell-checking system. In their study, they evaluated different methods for designing and implementing distinctive parts of the spell-checker, based on which the best decisions for applying on Persian language according to the most recent researches as well as the properties of Persian were made. In fact, they made use of syntactic analysis of the words and recognizing their lemma for better detection of errors and making appropriate suggestions. They also store the lexicon in the MDFA (Minimal Acyclic Deterministic Finite Automata) structure to reduce the size of the lexicon (Saburian and Dorri, 2006).

Also, Shamsfard and her colleagues in another study tried to use Web as a corpus in their method of spell-checking to modify previous classification offline methods (Shamsfard *et al.*, 2009).

Barari and Qasemizadeh presented an adaptive spell-checker for Persian language using a 'Ternary Search Tree' data structure, which is language-independent and built-in error pattern–free too. Their system detected misspelled words using non-deterministic traverse. That is, they solved the problem by traversing a tree with variable weighted edges. The system learns error pattern with some sample from language or media. It can adapt and tune itself by interactions by user or outer media and it improves its suggestion list as time goes by. The authors report that the proposed approach has more flexibility, accuracy, data compression rate, and reliability comparing with other methods. Moreover, it shows that it has an acceptable result for auto-selection problems (Barari and Qasemizadeh, 2005).

Kashefi and his colleagues implemented a spell-checker for Persian in which they used monolingual corpus for patterns analysis and occurrence rate of spelling errors. In the report of their study, they have not gone through details of the way their system works. Instead they focused more on the problems and challenges in Persian language processing and made some suggestions for tackling them. Their study also contains some appendices for numbers convertor, calendar convertor, Pinglish (Persian texts with English transcription) convertor, and modification of punctuation marks. Their proposed method is based on production and conjugation of verbs and adding them while loading and studying non-verb inflectional words (which are so many for every word in this language) without occupying memory for the retention. They evaluated their work using cross-validation with five-stage folding for the results to be more significant and valid and just (Kashefi *et al.*, 2010).

In this article, we present an alternative approach to spell-checking of Persian texts that uses a large corpus of Persian to detect the spelling errors and suggest a list of candidates among which the user can select the appropriate replacement of the misspelled word. As far as authors know, it is the first experience in making use of a corpus instead of a dictionary as a databank in spell-checking systems not only for Persian but for so many other languages. So, the methodology of this study can be well-applied to other languages too.

4 Methodology

4.1 Tokenization

In almost all spell-checking systems, the first step toward word identification is tokenization. Tokenization may refer to the division of the text into sentences and of sentences into words and punctuation. Sentence segmentation is not of any help in spell-checkers, whereas word segmentation is indispensable in such systems. Although punctuation marks such as '.', ',', '!', '?', ':', and ';' are used to separate sentences, they can also be used as word boundary besides white-space characters such as blanks, tabs, carriage returns, etc. Detecting errors based on predefined word boundaries is not problematic, except in cases where the running-on of two separate words and/or splitting a single word into two parts result in valid word(s) in the given language. The word 'forgot', for instance, may be written mistakenly as 'for got', splitting the word into two distinct ones, each of which can be found in the dictionary or corpus of the system as a valid word in English. Therefore, in such cases, tokenization cannot help detect the spelling errors or correct them. The only way for tackling this problem is to consider the context to which the erroneous word belongs.

4.2 Corpus preparation

For the purpose of this experiment we tried, as the first stage, to compile a comprehensive monolingual corpus of Persian texts consisting of >120 million tokens. By token we mean all individual members of a type in a text that may or may not be repetitive. The corpus is comprehensive in the sense that it has been divided into different sub-corpora of various text types, such as politics, medicine, poetry, sport, literature, art, religion, science, culture, history, economics, and miscellaneous. These texts are mainly extracted from books, journals, interviews, reports, written news, etc., but the main contribution goes with the online version of Hamshahri newspaper[2]. There were so many Persian books and papers in digital form having a variety of genres suitable for entering into the corpus. Many Persian Web sites contained interviews or reports on specific subjects from which we could extract required data for our corpus. In collecting data for the corpus, it has been tried to compile a balanced and representative corpus; i.e. it would contain texts from different domains and different genres in reasonable proportions to be a reasonable reflection of language use.

As for the other languages, the Persian language is being updated almost every day with new technical words and phrases, neologisms, loanwords, proper nouns, etc. For this reason, it is reasonable to use a large monitoring monolingual corpus as the source of our database used in a spell-checking system. The corpus and its accompanying database will be frequently building up and updating.

4.3 Persian lexicon generation

To construct a large-scale Persian lexicon to be used as a databank in the spell-checking system in this language, we have to tokenize the whole corpus in word level. Persian is among languages with a complicated morphological system, and there are many writing problems in this language due to some inconsistencies in different letter forms and space character between letters of a word. In this regard, a rather complicated algorithm has been designed and implemented to separate all tokens in the corpus. For example, there is a ZWNJ character in Persian used when writing some prefixes, suffixes, and compound words. Such non-printing character is used in case that it is desirable to keep two words closer together, which otherwise are either totally separated or connected to each other. Consider the following examples in this respect:

ZWNJ converts میخورم or می خورم, which are both incorrect, into the correct form می‌خورم

ZWNI converts لانه ها or لانهها, which are both incorrect, into the correct form لانه‌ها

ZWNI converts سیب زمینی, which is the incorrect form, into the correct form سیبزمینی

All these mentioned cases along with some other linguistic rules have been considered in the process of tokenization of the system.

After tokenization of the corpus, all tokens are to be converted into types with their relative frequencies in the corpus. The larger the size of the corpus in terms of types, the better and more precise the performance of the spell-checking system. This is especially true when one comes to deal with unknown words and/or proper nouns, as more and more words are added to the databank of the system when the size of the corpus is augmented. In the next step, the types such extracted are to be checked against the availability in Persian language. This is one of the most time-consuming and hard laboring tasks during the implementing of the system, which has been done manually. During this step, all garbage data as well as misspelled ones have been deleted from the corpus. This way, a database of around 1,000,000 word types was generated from the original corpus of Persian to be used in the system.

4.4 Error detection

4.4.1 *Dictionary lookup technique*

In dictionary lookup techniques, a program is run to check the whole dictionary to see if an input string appears in a dictionary or lexicon. If the string is not found, it is marked as a misspelled word. There are various techniques for dictionary lookup such as tries (Knuth, 1973), frequency ordered binary search tries (Knuth, 1973), and finite-state automata (Aho and Corasick, 1975). These standard search techniques have been used to reduce dictionary search time in different ways. A type of data structure program called *hash table* is the most common technique for fast access to dictionary or lexicon, in which for looking up a string in the lexicon, its hash address is computed and the string stored at that address is retrieved in a hash table, which has been previously constructed (Knuth, 1973). The technique for looking up the strings in the lexicon of the present spell-checker for Persian is based on the simple technique of indexing, which is more applicable for processing large data. All the words in the database were sorted alphabetically and then a string-matching algorithm (Navarro, 2001) was applied.

The size of the lexicon in a spell-checking system is a considerable factor in designing such systems. In fact, it has a direct relationship with the response time of search. That is, the larger the size or the number of dictionary entries, more time it takes to return the result of a search. For this very reason, many designers try to apply a lemmatizer to the lexicon of their spell-checking systems to reduce the volume of the system database. As a robust lemmatizer is not available for Persian, in this study, the word types extracted from our Persian corpus include all various paradigms of a word, i.e. singular and plural forms for nouns, different tenses for verbs, comparative and superlative forms of adjectives, and some other inflectional forms peculiar to Persian morphology. Although including all different forms of the words in the lexicon increases the size of the database to a high degree, we tried to apply a dynamic programming technique known as Levenshtein algorithm—a string-matching algorithm—(explained in section 4.5.1.) to deal with this problem.

Some scholars believe that the size of lexicon should be carefully decided on. A very small lexicon burdens the user with too many false rejections of valid terms, whereas a very large lexicon can result in an unacceptably high number of false acceptances (Kukich, 1992).

4.5 Candidate generation

Making proper suggestions for every error detected is the next stage in spell-checking systems, which should be carefully programmed. This stage can be divided into two sub-stages as: (1) generating a set of words that have a minimum edit distance to the erroneous words, and (2) selecting a subset of words extracted from the words produced in the first sub-stage, which exist in the database of the system. In fact, creating a letter adjacent matrix for Persian words, we tried to reduce the numbers of the candidate suggestions in the mentioned subset.

4.5.1 *Minimum edit distance*

The Damerau–Levenshtein distance, also known as minimum edit distance, is defined as the minimum number of editing operations (i.e. insertions, deletions, substitutions, and transposition) needed to convert one string into another (Wagner, 1974). In fact, in most spelling correction algorithms, the minimum edit distance between a misspelled word and a dictionary entry is to be computed. An erroneous word with a minimum distance of one from the correct word mostly results from pressing a key twice, typing two keys instead of one, skipping a key, and typing a key instead of another, and switching two adjacent letters in which the minimum edit distance is 2.

In this study, we used a dynamic programming technique known as Levenshtein algorithm with quadratic time complexity (Wagner and Fisher, 1974) to calculate the edit distance between a misspelled word and the words in database. Using this technique, it is not necessary for an erroneous input word to be compared with each word in the database to find proper candidates. In this respect, we passed every input word through an efficient algorithm to arrange a number of suggested words for each misspelled word. The algorithm in the following text demonstrates a set of steps to be taken to generate a number of candidates to be exposed to the users. In this algorithm, the minimum edit distance of the last editing operation is two, whereas the one in the other editing operations is one:

The Algorithm

Step 1: Take each input word (IW) that does not appear in the database as an erroneous one.
Step 2: Space between every two adjacent letters of IW.
Step 3: If any of the resulting words was found in the database as a valid word, return the result as a candidate word for the misspelled one.
Step 4: First join IW to the following word and then to the preceding one, and then repeat step 3.
Step 5: Substitute each letter in IW by all letters of Persian alphabet one by one, and then repeat step 3.
Step 6: Delete each letter in IW one by one, and then repeat step 3.
Step 7: Insert all letters of Persian alphabet one by one between each two adjacent letters and at the two ends of IW, and then repeat step 3.
Step 8: Exchange every two adjacent letters of IW, and then repeat step 3.

It should be mentioned that the two 'word boundary' steps (steps 2 and 3) should be performed before the regular order of the other steps.

4.5.2 *N-gram matrix generation*

N-grams are defined as n-letter subsequences of strings in which n is usually something between one and four and used in a variety of ways in text recognition and spelling correction techniques. Bigrams and trigrams matrices in this study are two means by which we can enhance the precision of our spell-checking system. Firstly, a matrix of every two adjacent characters of Persian alphabet, as well as a matrix of three adjacent ones, was compiled. These binary matrices consist of subsequent Persian characters along with their value as one or zero based on the information gained from the database depending on whether that string occurs in at least one word in a pre-defined lexicon. Blank character is also considered as a character in creating the two matrices. Consider, for instance, the strings چب (ch + b) and دپی (d + p + i), which are improbable in Persian language, whereas the strings چپ (ch + p) and دبی (d + b + i) are quite probable, and can be found in words such as چپاول (plunder) and دباغ (tanner). This way, we can check every two and three strings of a given word against these two n-gram matrices. If a string with a zero value is found anywhere in the given word, the program marks the string as an erroneous one and that string is regarded as the source of error in that word. Then, the minimum edit distance algorithm is to be applied only to the very string not to the whole word. As a result, the number of suggestions as well as the response time will be reduced in generating candidates for every misspelled word. Incorporating N-gram (letter adjacent) matrices to the minimum edit distance algorithm will inevitably increase the speed of the system performance. One of the studies that used N-gram matrices is a system presented by Golding and Schabes. Their system is based on trigrams that tackle the problem of correcting spelling errors, resulting in valid words (Golding and Schabes, 1996).

4.6 Ranking suggested candidates

In some spelling correction systems, the task of automatic correction is performed as a separate phase during which the system has the ability to automatically select the most suitable word as a substitute for the misspelled one among the suggested candidates generated from the previous phase. This task needs a kind of context analysis and intelligent learning from a large corpus with stochastic measures as well as some linguistic rules. At present, there is no such possibility for our spelling checker to be able to contextually analyze the monolingual corpus and automatically correct the detected errors. In fact, in the third phase, we managed to rank the candidate suggestions generated in the second phase according to their relative frequencies in the corpus. In most cases, the first priority, which is the one having the highest frequency in the corpus, is the most appropriate substitute of the erroneous word. Selecting from the suggested words thus sorted is up to the user.

5 The Experiment

To evaluate the performance of the present system, we carried out two sorts of evaluations. First, a test corpus of Persian texts consisting of sentences in which some words were artificially made erroneous with different lengths having one or two errors was prepared. The input texts contain ~10,000 words and the erroneous words 1,000 words. We tried to include all types of errors (deletion, insertion, substitution, exposition, and word boundary) in the test corpus with the reasonable proportion. In addition, a small set of misspelled words included proper nouns (especially proper names of persons), foreign words (those foreign words transcribed with Persian alphabet), neologisms (new words recently entered Persian language), and some technical terms (usually with a low occurrence frequency). In our second stage of evaluation, we tried to test the system on real data, a test corpus consisting ~1,000 Persian words extracted from one of the Persian weblogs about 'cupping' (hejamat) in traditional medicine. The results concerning the second evaluation scenario have been reported separately.

In the first phase, the text is analyzed word-by-word from right to left (according to Persian writing system). Each word that does not appear in the database is underlined and supposed to be a misspelled one. Here, it should be noted that nowhere in this experiment was a dictionary used. The database used in this experiment has been extracted from a large monolingual corpus of Persian containing >100 million tokens. The performance time, that is, the correction time as well as candidate generation and ranking time are the same as those in the real-time systems, regardless of the word length and the type of errors. This is one of the advantages of the present Persian spell-checking system, which makes it more practical than any other system. Figure 2 is an image of the environment in which our spell-checking system works. As the diagram shows, once a word was underlined as an erroneous one, a set of suggested words along with their relative frequencies in the corpus appear in a separate box. The first priority in this box, which has the highest frequency, is supposed to be the most probable substitution for the erroneous one. However, as selecting the first choice in the suggested list is not always the correct choice for the misspelled word, the Mean Reciprocal Ranking metric (suggested by one of the respected reviewers), which is suitable for applications in which only the first result matters, is used for evaluating the ranking procedures through the following formula:

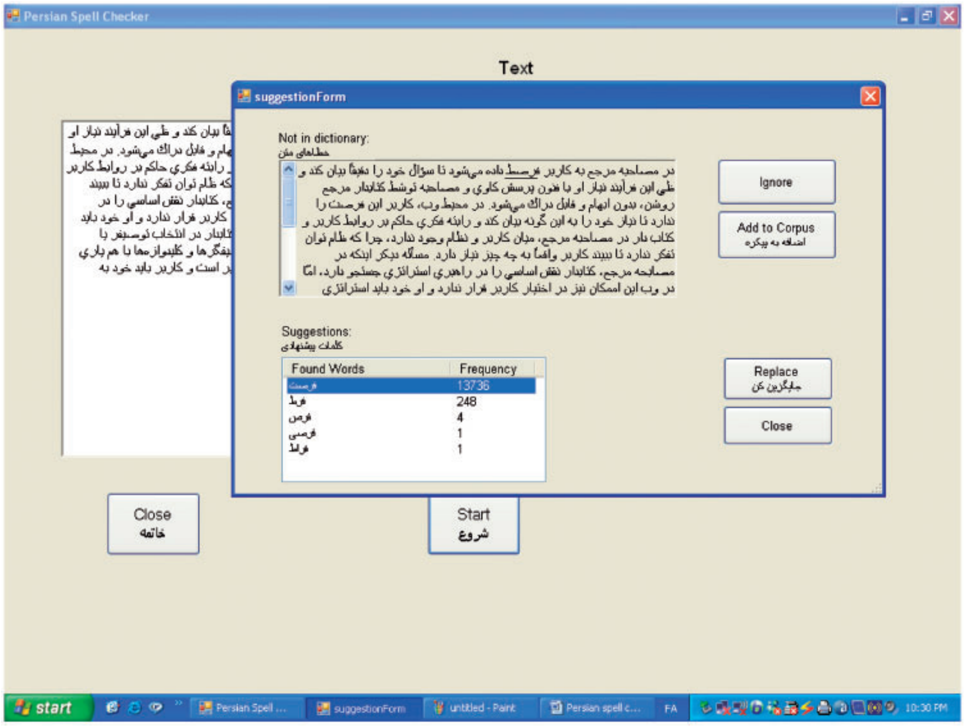
$$MRR = \frac{1}{|Misspellings|} \sum \frac{1}{Rank_{Correct\ Suggestion}}$$

If the underlined word is mistakenly recognized as an erroneous one, the user may click on 'add to corpus' button to allow the system to add it to the database. The 'ignore' button let the system go on checking the rest of the text without noticing the underlined word. The FarsiSpell has been implemented in visual C Sharp on a 2.3-GHz processor machine under Windows.

The third stage in the whole process of spell-checking, which aims at ranking a set of candidates from the most probable substitute to the least one, is not very accurate in sense that the ranking process is only based on frequencies, not the surrounding context. So, in some cases, the second, third or fourth candidates are selected instead of the first one, because all of them are deviated from the input word (erroneous word) only by one edit distance of insertion, deletion, substitution, or transposition transformations. Consider the misspelled word گزاش, shown in Table 1, for which all the mentioned candidates were displayed.

6 Results

The experiment results have been summarized in the tables and diagram. As Tables 2 and 3 show,



در مصاحبه مرجع به کاربر فرصط داده می‌شود تا سؤال خود را دقیقاً بیان کند و
طی این فرآیند نیاز او با فنون پرسش کاوی و مصاحبه توسط کتابدار مرجع
روشن، بدون ابهام و قابل درک می‌شود. در محیط وب، کاربر این فرصت را
ندارد تا نیاز خود را به این گونه بیان کند و رابطه فکری حاکم بر روابط کاربر و
کتاب دار در مصاحبه مرجع، میان کاربر و نظام وجود ندارد، چرا که نظام توان
تفکر ندارد تا ببیند کاربر واقعاً به چه چیز نیاز دارد. مسأله دیگر اینکه در
مصاحبه مرجع، کتابدار نقش اساسی را در راهبری استراتژی جستجو دارد. اما
در وب این امکان نیز در اختیار کاربر قرار ندارد و او خود باید استراتژی

Found Words	Frequency
فرصت	13736
فرط	248
فرص	4
فرصی	1
فرلط	1

Fig. 2 FarsiSpell system environment.

Table 1 A detailed report of the suggestion form for a misspelled word detected by Parspell system

Misspelled word	Candidates	Frequency	Edit distance	Transformation
گزاش	گزارش	104,165	1	Insertion
گزاش	گزاف	622	1	Substitution
گزاش	گزش	41	1	Deletion
گزاش	گواش	34	1	Substitution
گزاش	گازش	4	2	Transposition

our system works well on detecting erroneous words resulting from different sources. The only exception in this respect is 'word boundary' error type, for which the system does not present proper solution. The reason is, to some extent, that the tokenization part of the system is not sufficiently robust, and this, in turn, is due to different forms each Persian character takes depending on its position in a single word. In fact, the program detected all misspelled words but ~15–30% of word boundary errors. In the artificial test corpus, there were totally 123 word boundary errors, thirty-five were split words that were joined (e.g. ایـران and سل for ایـرانسل), and eighty-eight were joined words that were split (e.g. عرصههـای for عرصه and هـای). In the test corpus of real data, there were totally 207 word boundary errors, among which 176 cases were detected by the system.

The third column in each table represents the percentage of misspelled words for which the candidate suggestions have been ranked properly. As the first tables show, ~64% of the erroneous words in the artificial test corpus can be corrected automatically, whereas in the real test corpus, this quantity has been improved to up to >70%. In ranking the candidates' process, the mean reciprocal ranking metric has been used, which is more precise than calculating only the relative frequencies. In our work, we considered the top ten candidates suggested by the program to calculate the MRR.

Table 2 Different types of error found in the test corpus of artificial data

Error type	**Detected (%)**	**Correct decision**	**Example**
Word boundary	60.97%	43.20%	قابلتغییر=> قابل‌تغییر دانش مندان=> دانشمندان
Insertion	100%	85.1%	استثتنایی=> استثنایی
Deletion	100%	72.05%	امولشان=> اموالشان
Substitution	100%	56.9%	استسمار=> استثمار
Transposition	100%	61.8%	هزنیه => هزینه
Total	92.19%	63.81%	قابلتغییر=> قابل‌تغییر دانش مندان=> دانشمندان

Table 3 Different types of error found in the test corpus of real data

Error type	**Detected (%)**	**Correct decision**	**Example**
Word boundary	85.02%	53%	صرف نظر => صرف‌نظر عرضکرد=> عرض کرد
Insertion	100%	91%	صفیرا=> صفرا
Deletion	98.2%	77.5%	بپوشاید=> بپوشانید
Substitution	100%	60.6%	بلافاضله=> بلافاصله
Transposition	100%	70.51%	جحامت => حجامت
Total	**96.64%**	**70.52%**	

The idea behind MRR is that it multiplicatively inverses ~~the rank of the first candidate~~ the rank of the correct suggestion. If, for instance, the second candidate is the correct one, the MRR of that case is half. Similarly, if the third candidate happens to be the correct one, the MRR is one-third. The sum of all reciprocal ranks of the 1,000 erroneous words both on artificial data set and on real data set gained 0.76.

However, the output of this step is not precise and is subject to some modifications. With some improvements in tokenization of the system as well as taking the context into account in near future, we hope to increase automatic correction ability to a high degree.

As mentioned earlier, one of the advantages of the present system is making use of a large monitoring monolingual corpus of Persian as the basic database of the system instead of using a monolingual dictionary. A large corpus may contain so many non-dictionary words, such as proper nouns, neologisms, foreign words, etc. So, a database extracted from such a corpus is certainly richer than a dictionary. As Tables 4 and 5 show, the present system can detect >96% of non-dictionary words correctly and rank >95% of their suggested candidates properly. It goes without saying that some word classes overlap with each other. For example, the word هیستوگرام (histogram) belongs to both technical terms and loan words, or the word آلبرت belongs to both proper nouns and foreign words.

As far as the length of the misspelled words is concerned, detecting process works well when the

Table 4 Different classes of word found in the test corpus of artificial data

Word Class	**Detected (%)**	**Correct decision (%)**	**Example**
Proper nouns	87%	100%	جیرفتا => جیرفت
Foreign words	100%	100%	نیوکاشل => نیوکاسل
Neologisms	100%	50.8%	توفرم
Technical terms	94.1%	75%	سختافزار=> سخت افزار
Loan words	95.2%	70.2%	آنلیز => آنالیز
Regular words	92.3%	61.6%	مطبوات => مطبوعات

Table 5 Different classes of word found in the test corpus of real data

Word class	Detected (%)	Correct decision (%)	Example
Proper nouns	89.8%	98%	لـقـمـن –> لقمان
Foreign words	100%	100%	گـلـسـتـرول –> کـلـسـتـرول
Neologisms	98%	88.7%	پـزوهـانـه –> پژوهانه
Technical terms	100%	100%	فـسـد –> فصد
Loan words	100%	81%	مـاسـار –> ماساژ
Regular words	97.6%	78.5%	تـهـدیـدتـات –> تهدیدات

Table 6 Relationship between erroneous word length and system output

Word length	Detected (%)	Correct decision (%)	Example
N = 2	100%	33.3%	ات = مـات
N = 3	100%	44.5%	قـلـف => قـفـل
N = 4	100%	50%	قـمـت => قـیـمـت
N = 5	100%	78.2%	آلـبـریـت => آلـبـرت
N = 6	100%	82.7%	هـدف مـنـد => هـدفـمـنـد
N = 7	78.7%	88.6%	بـلـی اسـتـر => پـلـی اسـتـر
N = 8	68%	88.9%	کـامـیـپـوتـر => کـامـپـیـوتـر
N > 8	65%	34.1%	خشـونـتـآمـیـز => خشـونـت آمـیـز

number of characters in the word is less than seven. In fact, it is difficult for the system to detect errors in the long words. However, the proper ranking process of the candidates shows some improvement as the number of word characters grows. Although detecting errors in two- or three-letter words is completely successful, correcting them faces some problems due to the possibility of exchanging the characters in such words, resulting in other valid words. However, in most of them, the second or third priority is the correct decision. The results gaining based on the word length criterion is somehow similar in both kinds of evaluation performances. The relationship between the length of the erroneous words and the ability of the present system to detect the errors and select the right candidates are shown in Table 6.

Figure 3 clearly demonstrates that as the number of word characters (N) increases, the ability of the spell-checking program to correctly detect the errors decreases, whereas its ability to correctly rank the candidates for the misspelled word will increase (with the exception of $N > 8$). The difficulty of detecting errors in longer words seems to be due to their sparseness (rare occurrence) in the corpus. It means that, some of the long words or combined ones may not be available in the database of the system at all. Consider, for example, the Persian word بـحـث بـرانـگـیـزتـریـن (the most challenging), which is a rather rare one in Persian texts.

In this experiment, the result of the calculation of MRR for the overall performance of the system is 0.76, and the results of the evaluation metric of precision both on detection process and automatic ranking of the suggestions were considered and calculated as follows:

$$\text{Precision on detection} = \frac{\text{number of words whose correct form is ranked first}}{\text{number of words detected as being erroneous}}$$

$$\text{Precision on detection} = \frac{974}{1{,}032} = 94.3$$

$$\text{Precision on ranking candidates} = \frac{\text{number of words whose candidates ranked correctly}}{\text{number of words detected as being erroneous}}$$

$$\text{Precision on ranking candidates} = \frac{696}{1{,}032} = 67.4$$

7 Applications

A number of applications can be listed for which creating a rich database based on a large monitoring monolingual corpus and improvement in its abilities are of great benefit.

Text-to-speech or voice synthesis tools that make textual materials audible as well as speech recognition or speech-to-text tools will be more practical with some significant improvements in word recognition and correction. Machine translation and computer-aided language learning are other technologies; their performance will inevitably be

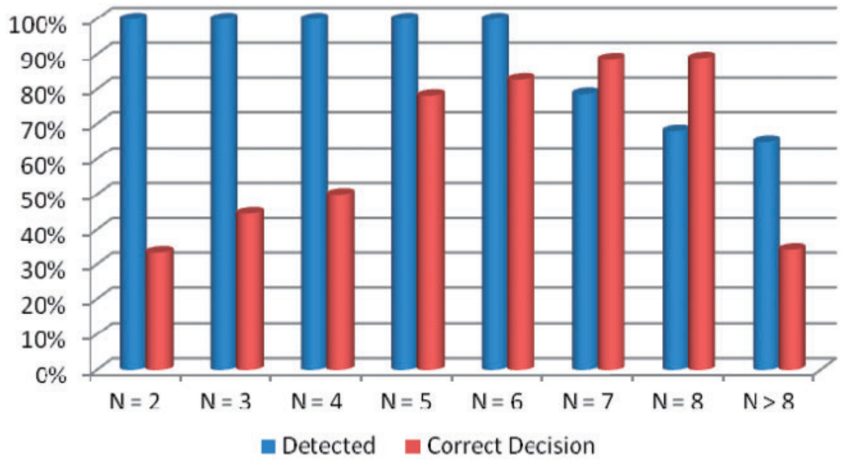


Fig. 3 The effect of word length on the system output.

affected by enrichment of such monolingual corpora. In fact, these applications need to be performed fully automatic with real-time capacity of word recognition and error correction.

OCR device is another technology making use of monolingual dictionaries to convert images of handwritten, typewritten, or printed text into machine-editable text. When a large monolingual corpus along with an n-gram matrix come to create the database behind an OCR device, its performance will inevitably be improved. English OCR devices have recently been developed with a high degree of accuracy. However, although Persian OCR devices have been developed a few years ago, they cannot practically cope with the problem of recognizing Persian characters in any sense (Salmani Jelodar *et al.*, 2005).

8 Conclusion and Further Development

In this study, it has been tried to introduce a novel system for checking misspelled words in Persian texts. The approach presented here is completely language-independent, and can be applied to any language that has a large monolingual corpus to create a language model.

The system of Persian spell-checker introduced by this study has some advantages against other similar systems designed for this language. In the following lines, it has been tried to compare our system with some others.

Sepanta Institute introduced a Persian spell-checker named ‘Vira’, in which a database of 100,000 entries of stemmed words along with >600 Persian root were used. According to its designers, it has the ability of detecting >1 million derivational words using stemming machine. Also, it is able to detect and correct spelling errors under Web using an additional Web service interface. Although this system has a proper speed in detecting and correcting errors, it does not have a proper support for Persian morphology and covers only the errors with edit distance of one. So, its outputs contain many false-positives dealing with compounds (SRRF, Vira, available at: http://www.spellchecker.ir/).

‘Virastyar’[3] is another spell-checker working as an additional feature to Microsoft Word for Persian users. Among its capabilities are detecting and correcting spelling errors, editorial mistakes, and punctuations, as well as Persian texts standardization. It has high efficiency with proper speed. However, it cannot work outside the Microsoft Word environment. Moreover, it seems that its database can be updated through user interaction and not automatically.

‘Vafa’[4] is a spell-checker software product presented by the researchers of the ‘Information and

Communication Technology Research Institute' of Iran. It has the ability of detecting and correcting spelling errors, grammatical errors (to some extent), as well as semantic ones. 'Vafa' system uses two Persian dictionaries (Dehkhoda and Moin) to deal with spelling errors, some grammatical rules to deal with grammatical errors, and statistical metrics such as Mutual Information and Confusion Sets dealing with semantic errors. Like 'Virastyar', it works only on Microsoft Word environment and does not have the ability to work as a separate module. One of the advantages of the proposed system in this study is its ability to use a database extracted from a large monolingual corpus of Persian, which is frequently updated through connecting to the Web. This way, even the most recent words and expressions may not be unregarded in its lexicon. Moreover, the system can work without any need to be installed to Microsoft Word. It acts as an isolated module working on any environment.

In FarsiSpell, the three steps common in most spell-checking systems, namely, detection of errors or misspelled words, during which the lexical analyzer detects the misspelled word in the input string; making suggestions for detected misspelled words, during which the system creates a set of possible candidates as potential replacement for the given erroneous word; and ranking candidates, during which these candidates are sorted out from the most likely replacements to the least likely ones, are all treated as a separate process and performed in sequence. However, the output of this step in our system is not precise and is subject to some modifications. With some improvements in tokenization of the system as well as taking the context into account in near future, we hope to increase automatic correction ability to a high degree. Given the context of every two adjacent words in the corpus, we can develop a context-sensitive method for spell-checking task, and the system will be able to go through the third step more accurately.

Making use of context in which every word may or may not occur in as well as calculating the Association Score (As) of every two words with a predefined window (Mosavi Miangah, 2008) along with a kind of shallow parsing throughout our monolingual corpus will certainly improve the performance of the present spell-checking system, enabling it to tackle errors resulting in valid words too. To perform accurate correction for such errors, both syntactic and semantic information from the surrounding context is to be incorporated to the system. Moreover, focusing on OCR-generated errors, we intend to develop a robust OCR device for Persian texts using facilities provided for the present experiment.

References

Aho, A. V. and Corasick, M. J. (1975). Fast pattern matching: an aid to bibliographic search. *Communications of the ACM*, **18**: 333–40.

Barari, L. and QasemiZadeh, B. (2005). Clonizer spell checker adaptive, language independent spell checker. In *Proceedings of the first ICGST International Conference on Artificial Intelligence and Machine Learning AIML*, **Vol. 5**. Cairo, Egypt, pp. 19–21.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, **7**: 171–6.

Ehsan, N. and Faili, H. (2011). Grammatical and context-sensitive error correction using a statistical machine translation framework. *Software: Practice and Experience*, **43**: 187–206.

Faili, H. (2010). Detection and Correction of Real-Word Spelling Errors in Persian Language. In *The 6th IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE'10)*, Beijing, China, August 2010, pp. 533–6.

Fossati, D. and Eugenio, B. D. (2007). A mixed Trigrams approach for context sensitive spell checking. *CICLing-2007, Eighth International Conference on Intelligent Text Processing and Computational Linguistics*. Mexico City, Mexico.

Golding, A. and Schabes, Y. (1996). Combining Trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th Meeting of the Association for Computational Linguistics*, Santa Cruz, CA, 24–27 June 1996, pp. 71–8.

Hassan, A., Noeman, S., and Hassan, H. (2008). Language independent text correction using finite state automata. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, Hyderabad, India, pp. 913–18.

Hodge, V. J. and Austin, J. (2003). A comparison of standard spell checking algorithms and a novel binary neural approach. *IEEE Transactions on Knowledge and Data Engineering*, **15**: 1073–81.

Johns, M. A., Story, G. A., and Ballard, B. W. (1991). Integrating multiple knowledge sources in a Bayesian OCR post-processor. In *Proceedings of IDCAR-91*. St Malo, France, 1991, pp. 925–33.

Kashefi, O., Nasri, M., and Kanani, K. (2010). *Towards Automatic Persian Spell Checking*. Tehran, Iran: SCICT.

Kolak, O. and Resnik, P. (2002). OCR error correction using a noisy channel model. In *Proceedings of the Second International Conference on Human Language Technology Research*. San Diego, CA, March 2002, pp. 257–62.

Knuth, D. E. (1973). *The Art of Computer Programming*, **Vol. 3**. (Sorting and Searching Algorithms). Reading, MA: Addison–Wesley.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, **24**: 377–439.

Mitton, R. (2010). Fifty years of spellchecking. *Writing Systems Research*, **2**: 1–7.

McIlroy, M. D. (1982). Development of a spelling list. *IEEE Transactions on Communications*, **COM-30**: 91–9.

Megerdoomian, K. (2000). *Persian Computational Morphology: A Unification-Based Approach*. Las Cruces, NM: Computing research Laboratory, New Mexico State University.

Mosavi Miangah, T. (2001). Comparative analysis of noun phrase for MT (with reference to English and Persian). *Problems of Language Theory and Translation Science, Moscow Pedagogical University*, **6**: 68–78.

Mosavi Miangah, T. (2008). Automatic term extraction for cross-language information retrieval using a bilingual parallel corpus. In *Proceedings of the 6th International Conference on Informatics and Systems (INFOS2008)*. Cairo, Egypt, 2008, pp. 81–4.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, **33**: 31–88.

Oommen, B. and Kashyap, R. (1998). A formal theory for optimal and information theoretic syntactic pattern recognition. *Pattern Recognition*, **31**: 1159–1177.

Pollock, J. J. and Zamora, A. (1984). Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, **27**: 358–68.

Rezvan, Y., Ivaz, K., and Rezvan, F. (1996). Towards spell checking in FarsiTeX. In *Proceedings of the 5th WSEAS International Conference on Data Networks, Communications and Computers*. Bucharest, Romania, 1996, pp. 248–50.

Saburian, M. and Dorri, S. (2006). Designing and implementing a Persian spell checker. In *2nd Workshop on Persian Language and Computer*. Iran, 2006, pp. 202–15.

Salmani Jelodar, M., Fadaeieslam, M. J., Mozayani, N., and Fazeli, M. (2005). A Persian OCR system using morphological operators. In *Proceedings of World Academy of Science, Engineering and Technology (WASET)*, **4**: 241–4.

Shamsfard, M., Kiani, S., and Shahedi, Y. (2009). STeP_1: standard text preparation of Persian Language. In *Proceedings of the Third Workshop on Computational Approaches to Arabic Script-based Languages (CAASL3)*. Ottowa, Canada.

Taylor, W. D. (1981). GROPE—a spelling error correction tool. *Bell Laboratories Technical Journal*.

Ullman, J. R. (1977). A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. *Computer Journal*, **20**: 141–7.

Wagner, R. A. (1974). Order-n correction for regular languages. *Communications of the ACM*, **17**: 265–8.

Wagner, R. A. and Fisher, M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, **21**: 168–78.

Yannakoudakis, E. J. and Fawthrop, D. (1983a). An intelligent spelling corrector. *Information Processing and Managemant*, **19**: 101–8.

Yannakoudakis, E. J. and Fawthrop, D. (1983b). The rules of spelling errors. *Information Processing and Managemant*, **19**: 87–99.

Zamora, E. M., Pollock, J. J., and Zamora, A. (1981). The use of trigram analysis for spelling error detection. *Information Processing and Managemant*, **17**: 305–16.

Zobel, J. and Dart, P. (1996). Phonetic string matching: lessons from information retrieval. *Presented at the Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. Zurich, Switzerland, 1996.

Notes

1. Optical Character Recognition
2. http://www.Hamshahrionline.ir
3. http://www.virastyar.ir/
4. http://www.vafaspellchecker.ir