A novel string distance metric for ranking Persian respelling suggestions

OMID KASHEFI, MOHSEN SHARIFI and BEHROOZ MINAIE

Natural Language Engineering / Volume 19 / Issue 02 / April 2013, pp 259 - 284
DOI: 10.1017/S1351324912000186, Published online:

Link to this article: http://journals.cambridge.org/abstract_S1351324912000186

How to cite this article:
doi:10.1017/S1351324912000186

Request Permissions : Click here
**A novel string distance metric for ranking Persian respelling suggestions**

OMID KASHEFI, MOHSEN SHARIFI* and BEHROOZ MINAIE

School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
e-mails: kashefi@{ieee.org, iust.ac.ir}, b_minaei@iust.ac.ir

(Received 8 April 2011; revised 16 June 2012; accepted 18 June 2012; first published online 24 July 2012)

Abstract

Spelling errors in digital documents are often caused by operational and cognitive mistakes, or by the lack of full knowledge about the language of the written documents. Computer-assisted solutions can help to detect and suggest replacements. In this paper, we present a new string distance metric for the Persian language to rank respelling suggestions of a misspelled Persian word by considering the effects of keyboard layout on typographical spelling errors as well as the homomorphic and homophonic aspects of words for orthographical misspellings. We also consider the misspellings caused by disregarded diacritics. Since the proposed string distance metric is custom-designed for the Persian language, we present the spelling aspects of the Persian language such as homomorphs, homophones, and diacritics. We then present our statistical analysis of a set of large Persian corpora to identify the causes and the types of Persian spelling errors. We show that the proposed string distance metric has a higher mean average precision and a higher mean reciprocal rank in ranking respelling candidates of Persian misspellings in comparison with other metrics such as the Hamming, Levenshtein, Damerau–Levenshtein, Wagner–Fischer, and Jaro–Winkler metrics.

1 Introduction

Error in spelling of words is inevitable (Stauffer 1949; Eastman and McLean 1981; Davis 1922) and that is why researchers have widely studied misspellings in different contexts (Damerau 1964; Hanson, Riseman and Fisher 1976; Abramovicic 1983; Pollock and Zamora 1983; Sterling 1983; Worthy and Viise 1996; Holmes and Malone 2004). Misspellings could be split into typographical and orthographical misspellings (Damerau 1964; Peterson 1986), or by another dichotomy could be split into conventional or consistent misspellings (Brown 1988).

Consistent misspelling of words occurs when the author does not know the correct spelling, is unsure about the alternate possibilities, or is convinced that the correct spelling of a word is a misspelling. In contrast, conventional misspelling of words

*Corresponding author: Email msharifi@iust.ac.ir
Table 1. Different single-error types (n.b. there is no correspondence between English and Persian words)

<table>
<thead>
<tr>
<th>Single-error types</th>
<th>English word (peace)</th>
<th>Persian word (صلح)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitution</td>
<td>peaxe</td>
<td>صاح</td>
</tr>
<tr>
<td>Omission</td>
<td>peac</td>
<td>صبح</td>
</tr>
<tr>
<td>Insertion</td>
<td>peasce</td>
<td>صلخ</td>
</tr>
<tr>
<td>Transposition</td>
<td>paece</td>
<td>صحل</td>
</tr>
</tbody>
</table>

occurs when the author knows the correct spelling of words but due to haste or inattention misspells them (Brown 1988).

On the other hand, orthographical errors result from errors in cognitive process when the author does not know the correct spellings of words. Orthographical misspellings could be phonemic, which are phonologically identical or very similar to the correct spellings, such as using *special* instead of *partial*, or could be non-phonemic, such as using *alternately* instead of *alternatively*. In contrast to orthographical errors, typographical errors are motoric errors that occur during typesetting. For example, a letter may be mistakenly deleted or inserted from/to a word (Ola 1973; Berkel and Smedt 1988).

More than 80% of typographical errors differ from correct spellings in the omission, substitution, or insertion of one single letter, or in the transposition of two adjacent letters (Damerau 1964; Peterson 1980a; Pollock and Zamora 1983). These types of errors are called single-errors (Damerau 1964; Peterson 1986), while errors caused by more than one of these four types of errors are called multi-errors (Kukich 1992). For example, we know four types of misspellings for the word *peace*, which means *صلح* in Persian (Table 1).

A traditional way to correct misspellings is to generate variations on misspellings, to look up each of them in a dictionary, to extract language words (Peterson 1980a), and to rank the extracted language words (Peterson 1980b; Mitton 2009). Different ranking strategies have been studied, including string and edit distance (Damerau 1964; Levenshtein 1966; Wagner and Fischer 1974; Yianilos 1983), probabilistic and statistical methods (Bledsoe and Browning 1959; Pollock and Zamora 1984), similarity keys (Odell and Russell 1918; Bledsoe and Browning 1959; De Heer 1982; Angell, Freund and Willett 1983; Zobel and Dart 1996; Holmes and McCabe 2002; Hodge and Austin 2003), and rule-based methods (Yannakoudakis and Fawthrop 1983; Means 1988). Researchers (Min, Wilson and Moon 2000; Mitton 2009) have shown that string distance and statistical methods, such as word frequency, are the best ranking strategies. However, neither of these two strategies is significantly superior to the other.

In this paper, we focus on a string distance strategy to rank spelling suggestions, and propose a string distance metric based on the character distance on the keyboard layout that takes into account the homophone and homomorph aspects of the Persian language. Homophone letters are letters that are pronounced the same, and
homomorph letters are letters that are not actually identical but are sufficiently similar to be confused. We show the superiority of our proposed string distance metric for Persian misspellings compared to other notable string distance metrics.

The rest of this paper is organized as follows. Section 2 presents the most related works on spell checking and string distance metrics. Section 3 presents an overview of the Persian language features. Section 4 outlines the Persian language spelling–error patterns derived from our analysis of large Persian corpora. Section 5 presents our proposed string distance metric in detail. Section 6 presents the evaluation of the proposed string distance metric and Section 7 concludes the paper.

2 Related work

There is a wealth of research on the detection and correction of misspelled words that have been reported by pioneers since early 1960s (Damerau 1964; Levenshtein 1966; Alberga 1967; Peterson 1980b; Pollock and Zamora 1983; Yannakoudakis and Fawthrop 1983) and more recently in the context of new approaches using machine-learning techniques (Jaro 1995; Golding 1996; Ristad and Yianilos 1998; Davrondjon and Janowski 2002; Rasooli, Kashefi, and Minaei 2011), n-grams (Ullman 1977; Golding and Roth 1999; Brill and Moore 2000), similarity and phonetic keys (Odell and Russell 1918; Zobel and Dart 1996; Holmes and McCabe 2002; Hodge and Austin 2003), and without using any dictionary (Peterson 1986; Comeau and Wilbur 2004). There has been however less research on the Persian language and similar languages such as Arabic and Urdu.

Naseem and Hussain (2007) have recently reported a good ranking method for the Urdu language. They have used hardcoded Shapex and Soundex codes for Urdu alphabets, and have proposed a hybrid ranking method by combining frequency statistics of words, Soundex code, Shapex code, and simple string distance. Although they have stated that statistical methods are not suitable for languages similar to Urdu, they have used statistics on frequencies of words and have presented a good error trend analysis without giving any comparative evaluation. The use of hardcoded Soundex and Shapex codes has led to better ranking, but extracting these codes for each language alongside their proof of correctness is a hard and costly task.

Shaalan, Allam and Gomah (2003) have thoroughly studied spelling errors such as homophones and homomorph letters in the Arabic language and have proposed an Arabic spell checker. Their work is limited to the finding of spelling errors and the provision of some suggestions. They have not presented any ranking process or evaluation of their work.

Shamsfard, Jafari and Ilbeygi (2010) have proposed STeP-1 as a set of fundamental tools for Persian text processing that includes a tokenizer, a morphological analyzer, a part of speech tagger, and a spell checker. Their proposed spell checker is just a space corrector and is limited to correcting inter-words spacing issues in Persian, and so is not really a spell checker. They have proposed a simple morphology analyzer, but a remarkably more comprehensive and accurate Persian morphology analyzer has been proposed by Megerdoomian (2000, 2004).
Since we have chosen to use a string distance strategy for ranking suggestions in the Persian language, here it seems reasonable to present some details on notable existing string distance metrics in particular as part of related works. There are many algorithms for defining and calculating the distance between two words. Some of these algorithms focus on the accuracy and the mechanism of string distance metric, such as the Hamming distance (Hamming 1950) and the Levenshtein distance (Levenshtein 1966), while other algorithms, such as the ones proposed by Hirschberg (1975), Masek (1980), and Ukkonen (1985), focus on improving the time complexity and the space complexity of the string distance metrics.

2.1 Levenshtein distance

The Levenshtein distance between two strings is the minimum number of operations required to transform one string into another string, where an operation is an insertion, deletion, or substitution of a single character (Levenshtein 1966). The Levenshtein distance can be applied to words with different lengths that are created either by insertion or omission of one or more letters. For example, the Levenshtein distance between \textit{language} and \textit{langouag} is 2 (i.e. 1 for the omission of ‘o’ and 1 for the insertion of ‘e’).

As it is shown in Algorithm 1, a function $f_i(i, j)$ is initially set to 0 for all $i$ and $j$ and $f_i(i, j)$ is then calculated for all query letters (query word is indicated by $q$ and query’s letters are indicated by $q_i$) and all lexicon-word letters (lexicon-word is indicated by $l$ and lexicon-word’s letters are indicated by $l_j$). The Levenshtein distance between $q$ and $l$ is iteratively calculated by comparing letters of $q$ and $l$, where each insertion, deletion, or substitution is awarded a score equal to 1, resulting in $f_i(|q|, |l|)$ that is the Levenshtein distance between $q$ and $l$.

**Algorithm 1** Levenshtein Distance

```
proc Levenshtein(q, l)
    for all $i$ where $j = 0$ and for all $j$ where $i = 0$ do $f_i(i, j) \leftarrow 0$
    if $q_i = l_j$
        then $d(q_i, l_j) \leftarrow 0$
    else $d(q_i, l_j) \leftarrow 1$
    $f_i(i, j) \leftarrow \min(f_i(i - 1, j), f_i(i, j - 1), f_i(i - 1, j - 1) + d(q_i, l_j))$
return $f_i(|q|, |l|)$
```

2.2 Damerau–Levenshtein distance

By analyzing very large text files, Damerau (1964) has found four types of typographical errors, namely insertion, deletion, substitution of a single character, and transposition of two adjacent characters. The Levenshtein distance only supports insertion, deletion, and substitution of letters. Thus, the Damerau–Levenshtein
A metric for ranking Persian respelling suggestions

Distance can be considered as an extension to the Levenshtein distance that includes the added cost of transposition of two adjacent characters (Algorithm 2).

**Algorithm 2** Damerau–Levenshtein Distance

```plaintext
proc DamerauLevenshtein(q, l)
for all i where j = 0 and for all j where i = 0 do fdl(i, j) ← 0
if qi = lj then d(qi, lj) ← 0
else d(qi, lj) ← 1
if (qi = lj−1) and (qi−1 = lj) then t(qi, lj) ← 1
else t(qi, lj) ← 2 /*Relatively Large Value*/
fdl(i, j) ← min(fd(i−1, j), fd(i, j−1), fd(i−1, j−1) + d(qi, lj), fd(i−2, j−2) + t(qi, lj))
return fd(|q|, |l|)
```

**2.3 Wagner–Fischer Distance**

The Wagner–Fischer algorithm is a modified version of the Levenshtein distance algorithm that considers different costs for insertion, deletion, and substitution (Wagner and Fischer 1974). This algorithm is similar to the Levenshtein and Damerau–Levenshtein algorithms in preserving the \(O(mn)\) for time and space complexity, where \(m\) is the length of word \(q\) and \(n\) is the length of word \(l\). The Wagner–Fischer algorithm given in Algorithm 3 is similar to the Levenshtein algorithm with the exception that Levenshtein uses 1 for the cost of insertion or deletion whereas Wagner–Fischer uses different costs for insertion and deletion of letters.

**Algorithm 3** Wagner–Fischer Distance

```plaintext
proc WagnerFischer(q, l)
for all i where j = 0 and for all j where i = 0 do fwf(i, j) ← 0
d(qi, ε) ← cost of deletion
d(ε, lj) ← cost of insertion
d(qi, lj) ← cost of substitution
fwf(i, j) ← min(fwf(i−1, j) + d(qi, ε), fwf(i, j−1) + d(ε, lj), fwf(i−1, j−1) + d(qi, lj))
return fwf(|q|, |l|)
```

**2.4 Jaro–Winkler distance**

The Jaro–Winkler distance algorithm (Jaro 1989, 1995) is a variant of the Jaro distance algorithm that has been mainly used in record linkage. Algorithm 4 shows the Jaro distance between strings \(q\) and \(l\). The Jaro distance is scaled into a
normalized score where 0 means non-similarity and 1 means an exact match. The Jaro–Winkler distance algorithm (Winkler and Thibaudeau 1991; Winkler 1999) modifies the Jaro distance and adds a factor of common prefix at the start of the string as it is shown in Algorithm 5. This score is normalized too, and 0 represents inequality and 1 represents an exact match (Winkler 1999).

**Algorithm 4 Jaro Distance**

```
proc Jaro(q, l)
    m ← number of matching letters
    t ← number of transpositions of adjacent letters
    fj ← \frac{1}{3} (\frac{m}{|q|} + \frac{m}{|l|} + \frac{m-t}{m})
return fj
```

**Algorithm 5 Jaro–Winkler Distance**

```
proc JaroWinkler(q, l)
    l_c ← max(length of initial common prefix(q, l), 4)
    \rho ← 0.1 /*Scaling Factor*/
    fjw ← Jaro(q, l) + l_c \times \rho \times (1 - Jaro(q, l))
return fjw
```

### 2.5 Hamming distance

The Hamming distance of two equal length words, as it is shown in Algorithm 6, is the number of positions that have different letters (Hamming 1950). For example, the Hamming distance between write and while is 2. To convert this score to a normalized score (from 0 to 1), the score is divided into the word’s length such that the normalized Hamming distance between write and while is 2/5. For strings with different lengths, a modified version of the Hamming distance is used that increases the distance by one point for each additional character. For example, the Hamming distance between see and season is 4 (1 for mismatching third letters of words, and 3 for son) and the normalized score is 4/6.

**Algorithm 6 Hamming Distance**

```
proc Hamming(q, l)
    fh ← 1
    sum ← 0
    for i = 0 to max(|q|, |l|) do
        if qi \neq lj
            then sum ← sum + 1
    return fh = \frac{sum}{max(|q|, |l|)}
```
So far we have studied several spell checking approaches to Persian and Persian-like languages and outlined the respelling ranking strategies in this section. As we have chosen in this paper to focus on the string distance strategy to rank the respelling suggestions, we also explained in detail some notable string distance metrics, namely the Hamming, Levenshtein, Damerau–Levenshtein, Wagner–Fischer, and Jaro–Winkler distance metrics.

3 Persian language attributes

The Persian language has a rich morphology (Megerdoomian 2000). Persian words can be combined with a very large number of affixes. The Persian language is highly derivational, but combination, derivation, and inflection rules are uncertain (Mahootian and Gebhardt 1997; Lazard 2012). Word spacing is complex and different from other languages. Many Persian alphabet letters are homophone and homomorph. All these features affect the Persian spelling, so in this section we study the Persian language features that must be taken into account in the context of spell checking.

3.1 Homophones

A homophone is a word that is pronounced the same as another word. Homophones may be spelled the same (homograph) such as rose (flower) and rose (past tense of rise), or differently (heterograph) such as carat and caret. Homophones may be a correct language word with different meanings similar to homographs, or a misspelled word similar to some heterographs such as karet instead of caret. Our focus in this paper is on misspelled homophones. Those who try to write in languages other than their native language often make homophone misspellings. The Persian language has a large number of letters that are pronounced the same but are heterograph. Such letters increase the amount of homophone misspellings even by educated Persians. For example, the word /daɣdæye/ meaning concern can be misspelled as دادگاه, دادگه, دادیگه, دادگی, دادگه, دادیگه, or دادیگه. We have categorized the Persian homophone letters into seven classes (Table 2).

Table 2. Persian homophone letters

<table>
<thead>
<tr>
<th>Group</th>
<th>Members</th>
<th>Common IPA</th>
<th>Other IPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alef</td>
<td>{א}</td>
<td>/a/</td>
<td>/æ/, /e/, /o/, /?/</td>
</tr>
<tr>
<td>Teh</td>
<td>{ת}</td>
<td>/t/</td>
<td></td>
</tr>
<tr>
<td>Hamza</td>
<td>{ه، ق، گ، ا، ا، ط، ا}</td>
<td>/ʔ/</td>
<td>/æ/, /e/, /o/, /j/, /y/, /u/, /i/, /a/</td>
</tr>
<tr>
<td>Seen</td>
<td>{س، ص، س}</td>
<td>/s/</td>
<td></td>
</tr>
<tr>
<td>Zain</td>
<td>{ز، ض، ض}</td>
<td>/z/</td>
<td></td>
</tr>
<tr>
<td>Heh</td>
<td>{ه، ح}</td>
<td>/h/</td>
<td></td>
</tr>
<tr>
<td>Qaf</td>
<td>{ق، غ}</td>
<td>/ɣ/, /g/</td>
<td></td>
</tr>
</tbody>
</table>
Table 3. Persian homomorph letters

<table>
<thead>
<tr>
<th>Members</th>
<th>IPA (in the reverse order of appearance in the set)</th>
<th>Homomorph forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>{أ، أ}</td>
<td>/a/, /æ/</td>
<td>All forms</td>
</tr>
<tr>
<td>{پ، پ}</td>
<td>/p/, /b/</td>
<td>All forms</td>
</tr>
<tr>
<td>{س، س}</td>
<td>/s/, /t/</td>
<td>All forms</td>
</tr>
<tr>
<td>{ح، ح، ج، چ}</td>
<td>/ʃ/, /dʒ/, /h/</td>
<td>All forms</td>
</tr>
<tr>
<td>{خ، خ}</td>
<td>/x/, /h/</td>
<td>All forms</td>
</tr>
<tr>
<td>{ض، ص}</td>
<td>/z/, /s/</td>
<td>All forms</td>
</tr>
<tr>
<td>{ط}</td>
<td>/z/, /t/</td>
<td>All forms</td>
</tr>
<tr>
<td>{غ، ع}</td>
<td>/ɣ/, /α/, /ʔ/</td>
<td>All forms</td>
</tr>
<tr>
<td>{گ، گ}</td>
<td>/g/, /k/</td>
<td>All forms</td>
</tr>
<tr>
<td>{ق، ق، چ، چ}</td>
<td>/χ/, /α/, /ʔ/</td>
<td>Initial and medial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mitton (2009) has shown that the calculation of traditional string distance is not quite effective for homophones. He has reported that in more than 56% of cases the best suggestion is a homophone of misspelling if the homophone is in the candidates list. Given the large number of homophone letters in the Persian language, the probability of homophones being selected as best suggestions will be even more.

### 3.2 Homomorph

Most Persian letters have four different forms depending on their positions in a word: isolated form (e.g. /gAf/ in word /mærg/ meaning death), initial form (e.g. /gærm/ meaning warm), medial form (e.g. /tʃæng/ meaning fork), and final form (e.g. /ræng/ meaning color). Some Persian letters resemble each other in all or most of their forms. Table 3 shows the homomorph letters of the Persian language. These letters are hard to differentiate even during proofreading, especially when small size fonts or fonts with poor designs are used. Furthermore, because most homomorph letters are adjacent on the Persian keyboard layout, the probability of mistakenly using one letter in place of another homomorph letter is quite high.

### 3.3 Diacritics

A diacritic is an additional glyph attached to a letter or a basic glyph that can affect the pronunciation of a letter or the meaning of a word, or both (Korpela 2006). Persian script, which is based on the Arabic alphabet, consists of three main diacritics, including Harekat that represents short vowel marks (i.e. یـــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــــ~

In the Persian script, the use of Harekat is optional but Tashdid and Tanvin are mandatory and their absence may cause misspelling. However, diacritics are rarely used in Persian scripts. For example, people write
/ræsmaen/ meaning officially as mason, or they write /bænA/ meaning building, and unfortunately pronounce them the same, use them the same, and consider them the same.

4 Persian spelling error pattern

The analysis of typographical error patterns demonstrates the probability of each major typographical error, namely insertion, omission, substitution, and transposition. Several studies on the patterns of spelling errors in English have been reported, the most notable of which have been carried out by Damerau (1964), Pollock and Zamora (1983), Peterson (1986), and Mitton (2009) using large English corpora. To study the pattern of Persian spelling errors, we have used three sources of data: Hamshahri corpus (AleAhmad et al. 2009), CRCIS books, and undergraduate and postgraduate students' thesis publications in the School of Computer Engineering of the Iran University of Science and Technology (SCE-IUST). The Hamshahri corpus was extracted from articles published from 1996 to 2002 in the Iranian Hamshahri daily newspaper. The corpus consisted of 1,60,000 articles in eighty-three categories with 15 million words (4,17,000 distinct words). Since the typography of newspaper articles (at least in Iranian press) is rarely proof checked, almost every article includes spelling errors. Thus, the rate of misspelling in the Hamshahri corpus is relatively high. CRCIS corpus was built from more than 800 books with 50 million words (3,98,000 distinct words). CRCIS is a professional publisher; two independent typists type every book; by comparing these two versions, the rate of spelling errors in the final version of the book is considerably low. Diacritics are taken care of too. SCE-IUST corpus was a collection of students' thesis documents from 2006 to 2009. This corpus included ninety documents on Computer Science and Engineering containing 2.1 million words (2,31,000 distinct words). Thesis are mostly peer-reviewed and students are familiar with computational challenges of the language, so the rate of spelling errors in a thesis is not high. We have made a cumulative corpus (hereinafter called our corpus) from the aforementioned three corpora. Our corpus contained nearly 67 million words (5,00,000 distinct words). Table 4 shows the Persian spelling error patterns in each corpus.

We automatically extracted the misspellings, manually proof checked them, and found the correct respelling suggestion of each one. The total number of misspellings was about 40,000 words (0.06%). About 88% of these misspellings were single errors, which were either the insertion or omission of one single letter, substitution of one letter by another, or transposition of adjacent letters. The rest of misspellings (12%) were multiple-errors, which were caused by more than one single error. Substitution of letters accounted for most of the single errors (52%), and the rest of the single errors were marshaled as omission of a letter (27%), insertion of a letter (15%), and transposition of two adjacent letters (6%). Most of the multiple errors (92%) had just two single errors. Considerable amount of misspellings (19%) were homophones of correct words; mostly (82%), with just one single letter difference. Large amounts of misspellings (34%) were caused by the substitution of homomorph letters. Words that should have been written with diacritics constitute a minute part of Persian words, but we observed that they were mostly (88%) misspelled with no diacritics.
### Table 4. The pattern of Persian spelling errors

<table>
<thead>
<tr>
<th></th>
<th>Single-error</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Multiple-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Substitution</td>
<td>Insertion</td>
<td>Omission</td>
<td>Transposition</td>
<td>Single</td>
<td>Multiple</td>
</tr>
<tr>
<td>Hamshabri</td>
<td>62%</td>
<td>92%</td>
<td>19%</td>
<td>15%</td>
<td>4%</td>
<td>79%</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>17,327 (82%)</td>
<td>3,832 (18%)</td>
<td></td>
<td></td>
<td></td>
<td>4,304 (20%)</td>
<td>8,091 (38%)</td>
</tr>
<tr>
<td></td>
<td>21,159 (100%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRCIS</td>
<td>43%</td>
<td>97%</td>
<td>36%</td>
<td>15%</td>
<td>6%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>7,249 (95%)</td>
<td>391 (5%)</td>
<td></td>
<td></td>
<td></td>
<td>1,066 (14%)</td>
<td>2,572 (34%)</td>
</tr>
<tr>
<td></td>
<td>7,620 (100%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IUST-CE</td>
<td>44%</td>
<td>90%</td>
<td>33%</td>
<td>14%</td>
<td>9%</td>
<td>84%</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td>12,291 (91%)</td>
<td>1,182 (9%)</td>
<td></td>
<td></td>
<td></td>
<td>2,532 (19%)</td>
<td>3,682 (27%)</td>
</tr>
<tr>
<td></td>
<td>13,473 (100%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cumulative</td>
<td>52%</td>
<td>92%</td>
<td>27%</td>
<td>15%</td>
<td>6%</td>
<td>82%</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>34,809 (88%)</td>
<td>4,910 (12%)</td>
<td></td>
<td></td>
<td></td>
<td>7,687 (19%)</td>
<td>13,695 (34%)</td>
</tr>
<tr>
<td></td>
<td>39,719 (100%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Number of all misspellings.
* Number of all words that should have been written with diacritics.

Researchers have reported that the first letter in English is usually correct (Pollock and Zamora 1983; Yannakoudakis and Fawthrop 1983; Mitton 1987). In contrast, our analysis of Persian has shown that only 18% of misspellings were caused by mistakes in the first letter of a word. Given that we used an average length of 6 for Persian words in our analysis, we can conclude that the probability of misspelling caused by erroneous first letter of a word is the same as the probability of misspelling of other letters of the word.

### 5 The proposed string distance metric

We mentioned the most important features of the Persian language that must be considered for spell checking such as as homophones, homomorphs, and diacritics.
Referring to our studies on related works in Section 2, we did not find any notable work that considers homophones, homomorphs, and keyboard layout effects altogether. Furthermore, those works that have considered phonemic aspects of languages, have not reported considerable correction precision (Pollock and Zamora 1984; Kukich 1992; Zobel and Dart 1996; Hodge and Austin 2001; Holmes and McCabe 2002; Naseem and Hussain 2007).

In this section, we present a novel string distance metric considering homophones, homomorphs, diacritics, and keyboard layout effects. We consider four major single-error types, namely insertion, omission, substitution, and transposition. We present a statistical method to determine the cost for each mentioned single-error type by considering the effects of keyboard layout on typographical types of misspellings. In addition, we present a simple but effective method to satisfy homophone and homomorph types of misspellings.

### 5.1 Single-error distance

In Section 4, we statistically analyzed large Persian corpora for distribution of different single-error types. Taking a difference in distribution as a different probability in the occurrence of each type of single-error, we can define the probability of occurrence of each type of single-error by (1).

\[
P_{SE}(t) = \frac{\text{Total Number of Misspellings Caused by Single Error } t}{\text{Total Number of Misspellings}} \tag{1}
\]

Suggestions that are different from misspelling by more probable single-error types are themselves more probable to be the best suggestion. Therefore, the best suggestion must have less distance with misspelling caused by more probable single-error. We can thus define the distance of each single-error type by (2).

\[
Distance_{SE}(t) = \sum P_{SE} - P_{SE}(t) \tag{2}
\]

Table 5 shows the results of our studies on the probability of occurrence of each type of single-error and the distance cost of single-errors. Among distances presented in Table 5, substitution distance and omission distance require special consideration. Keyboard layout, homophones, and homomorphs affect the substitution distance
of two letters. The omission of diacritics must be considered differently from the omission of letters.

5.2 Substitution distance

The distance of substitution of two letters could be different when two substituted letters are homophone or homomorph than when they are not. In addition, especially in typographical errors, the substitution distance of two letters that are closer to each other in the keyboard layout are not the same as those that are far from each other. In this section, we propose a substitution distance function that considers the effect of keyboard layout, homophones, and homomorphs.

5.2.1 The keyboard layout effect

Substitution of letters is affected by the position of each character on the keyboard such that two adjacent characters on the keyboard are more likely to be substituted by each other compared to two letters that are more distant (Eastman and McLean 1981); this distance is called the character distance. For more clarification, Figure 1 shows an English QWERTY keyboard layout, and Figure 2 shows the widely used Persian standard keyboard layout.

The character distance between two characters $c_1$ placed on $(x_1, y_1)$ and $c_2$ placed on $(x_2, y_2)$ is calculated using the Euclidean Distance (Heath 1956). As an example, the distance between character ‘a’ placed on (0.5, 1) and the character ‘p’ placed on (9, 2) is $\text{Euclidean Distance}(p, a) = \sqrt{(9 - 0.5)^2 + (2 - 1)^2} = 8.5586$. Similarly, the distance between character ئ /za/ placed on (1, 0) and the character ئ /t/ placed on (11, 2) is $\text{Euclidean Distance}(\text{ئ}, \text{ج}) = \sqrt{(11 - 1)^2 + (2 - 0)^2} = 10.1980$.

1 The most recent but rarely used standard for the Persian keyboard layout is ISIRI 9174, see: http://std.isiri.org/std/9147.pdf.
Table 6. Statistics of misspellings involving upper characters in Persian

<table>
<thead>
<tr>
<th></th>
<th>Hamshahri</th>
<th></th>
<th>CRCIS</th>
<th></th>
<th>IUST-CE</th>
<th></th>
<th>Cumulative</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
<td>Multiple</td>
<td></td>
<td>Single</td>
<td>Multiple</td>
<td></td>
<td>Single</td>
<td>Multiple</td>
</tr>
<tr>
<td></td>
<td>98%</td>
<td>2%</td>
<td></td>
<td>100%</td>
<td>0%</td>
<td></td>
<td>98%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>384 (100%)</td>
<td>2% of 21,159&lt;sup&gt;a&lt;/sup&gt;</td>
<td>274 (100%)</td>
<td>4% of 7,620&lt;sup&gt;a&lt;/sup&gt;</td>
<td>107 (100%)</td>
<td>0.8% of 13,473&lt;sup&gt;a&lt;/sup&gt;</td>
<td>512 (100%)</td>
<td>1% of 39,719&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>349 (91%)&lt;sup&gt;c&lt;/sup&gt;</td>
<td>4% of 9,420&lt;sup&gt;b&lt;/sup&gt;</td>
<td>188 (76%)&lt;sup&gt;c&lt;/sup&gt;</td>
<td>8% of 3,446&lt;sup&gt;b&lt;/sup&gt;</td>
<td>103 (96%)&lt;sup&gt;c&lt;/sup&gt;</td>
<td>421 (82%)&lt;sup&gt;c&lt;/sup&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup> Number of all misspellings.
<sup>b</sup> Number of all words that should have been written with upper character(s).
<sup>c</sup> Number of times a word with corresponding upper characters is the correct suggestion if it exists in the suggestions list.

Therefore, we define the **Character Euclidean Distance** of two characters considering the keyboard layout by (3) that yields a normalized value from 0 to 1, indicating that lower distances have lower scores,

\[
\text{Character Euclidean Distance}(q, l) = \frac{\text{Euclidean Distance}(q, l)}{\text{Maximum Euclidean Distance}}
\]

**Upper characters.** In the Persian keyboard layout, the typing of some characters requires the pressing and holding of the *shift key* before hitting the characters on the keyboard. These characters are called *upper characters* and are separated by a slash sign (‘/’) in Figure 2, such as $\epsilon_3/ /z/ \text{ or } \epsilon_4/ /a/$. The distance between upper characters and other characters cannot be calculated simply by using the Euclidean distance. Note that failing to press the shift key or using inadequate force to press the shift key when typing an upper character is more probable than mistakenly pressing the shift key for typing a non-upper character. As an example, it is more probable that a user types $\epsilon_3/ /ze/ \text{ by mistake instead of } \epsilon_2/ /e/$ instead of $\epsilon_3 / \text{ by mistakenly pressing the shift key. The same is true for the distances between upper characters and other characters either upper or regular.}

Table 6 includes the details of our analysis on misspellings involving upper characters. In approximately 80% of misspellings involving upper characters, the correction differed from the misspelling by the substitution of a lower character for a corresponding upper character (see bottom right of the Table 6). In addition, respectively large amount of upper character misspellings (4%) in contrast with regular misspellings (0.06%, see Section 4) indicates that upper characters’ effects on misspellings must be considered individually.

The character distance of a mistakenly used lower character from an equivalent upper character must be less than the minimum Euclidian distance on the keyboard layout. This is because a character on the keyboard layout has 2 to 4 (left, right,
up, and down adjacent characters) replacement candidates with minimum Euclidian distance. If there are four possibilities, then each of them has a 25% chance of being the correct replacement; and with two possibilities, each of them has 50% chance. In either case, it is far less than 80%.

Algorithm 7 shows our proposed metric for calculating the character distance considering upper characters between $c_1$ and $c_2$, where $c_1$ is a misspelled character and $c_2$ is a replacement candidate. When $c_1$ and $c_2$ are either a lower or an upper character, we set their distance to their Euclidian distance in the keyboard layout. However, when $c_1$ is an upper character and $c_2$ is a lower character, we increase their Euclidian distance by the Euclidian distance between $c_2$ and the shift key, i.e. when actually the shift key is pressed by mistake instead of $c_2$. Later in Algorithm 8, we bias the distance with substitution cost (i.e. $Distance_{se}(\text{substitution})$ calculated by (2) and presented in Table 5), so the character distance between an upper character and its corresponding lower character would well match the substitution cost range.

**Algorithm 7 Proposed Character Distance**

```
proc ProposedCharacterDistance(c_1, c_2)
    \[ d_{\text{max}} \leftarrow \text{Maximum of Characters' Euclidean Distance on Keyboard Layout} \]
    \[ d_{\text{avg}} \leftarrow \text{Average of Characters' Euclidean Distance on Keyboard Layout} \]
    \[ \text{if } (c_1 \text{ is a Lower Character or both } c_1 \text{ and } c_2 \text{ are Upper Character}) \]
    \[ \text{then } cost_s \leftarrow \text{Character Euclidean Distance}(c_1, c_2) \quad (*\text{From Equation (3)}*) \]
    \[ \text{else } cost_s \leftarrow \text{Euclidean Distance}(c_1, c_2) + \text{Euclidean Distance}(c_2, \text{Shift Key}) \]
    return cost_s
```

5.2.2 Attributes of homophones and homomorphs

As we mentioned in Section 3.1, the Persian language has a large number of homophone letters that can cause a large number of misspellings with the same pronunciation as the correct word (homophone misspellings). Also, referring to our studies on Section 4 about the patterns of the Persian spelling errors, homophone misspellings constitute a considerable amount (19%) of the Persian language’s misspellings. Homophone misspellings mostly occur because of the substitution of a letter by another homophone letter, which mostly is not adjacent or even near to it in the Persian keyboard layout. Therefore, the sole use of the proposed character distance based on the keyboard layout may not be effective for ranking homophone misspellings.

Homomorphs, like homophones, need special consideration. Homomorph letters are commonly used interchangeably such as when a novice typist searches for the position of a letter on a keyboard, or when digital texts are recognized by Optical Character Recognition (OCR) techniques. In addition, some professional typists, who type handwritten drafts, type the texts as they look without giving much attention to their meanings, so homomorph letters may well be used in place of one another (Salthouse 1986; Naseem and Hussain 2007). The Persian homomorph letters are usually placed adjacent to each other on the Persian standard keyboard layout.
Table 7. The probability of occurrence and the distance of homophones and homomorphs in Persian

<table>
<thead>
<tr>
<th>Substitution error type</th>
<th>Portion in substitution</th>
<th>Probability</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homomorph</td>
<td>27.54%</td>
<td>0.275</td>
<td>0.138</td>
</tr>
<tr>
<td>Homophone</td>
<td>17.67%</td>
<td>0.177</td>
<td>0.236</td>
</tr>
</tbody>
</table>

and thus their substitution costs using our proposed character distance is small. Nevertheless, exceptions, such as Hamza homomorph letters grouped in Table 3, may not be placed near each other on the keyboard layout.

Given the considerable amount of homophone and homomorph misspellings, we differentiate the homophone and homomorph distances from the regular substitution distance (presented in Table 5). We calculate the probability of the occurrence of single homophone and homomorph errors in single substitution errors. We have used (1) and (2) but this time for single substitution errors, and not for total number of misspellings. Table 7 shows our studies on the probability of the occurrence and the distances of single homophone and homomorph misspellings.

Algorithm 8 shows the proposed substitution distance considering the keyboard layout, homophones, and homomorphs. The ProposedCharacterDistance is a normalized value from 0 to 1. Therefore, we adjust the value to match the substitution distance presented in Table 5. The substitution distance of two letters, which are not homophone or homomorph, is from \( \text{Distance}_{s}(\text{substitution}) \) to 1.

\[
\text{Algorithm 8 Proposed Substitution Distance of Two Letters}
\]

\[
\text{proc } \text{SubstitutionDistance}(c_1, c_2) \\
\qquad \text{if } (c_1 \in \text{HomophoneFamily}(c_2)) /*\text{See Table 2}*/ \phantom{c_1} \\
\qquad \quad \text{cost}_s \leftarrow \text{Distance}_{s}(\text{homophone}) /*\text{From Equation (2) and Table 7}*/ \phantom{c_1} \\
\qquad \quad \text{if } (c_1 \in \text{HomomorphFamily}(c_2)) /*\text{See Table 3}*/ \phantom{c_1} \\
\qquad \quad \quad \text{cost}_s \leftarrow \text{Distance}_{s}(\text{homomorph}) /*\text{From Equation (2) and Table 7}*/ \phantom{c_1} \\
\qquad \text{else} \phantom{c_1} \\
\qquad \quad \text{cost}_s \leftarrow \text{ProposedCharacterDistance}(c_1, c_2) \times (1 - \text{Distance}_{s}(\text{substitution})) + \text{Distance}_{s}(\text{substitution}) \phantom{c_1} \\
\text{return } \text{cost}_s \\
\]

5.3 Omission distance

In Section 4, we analyzed the pattern of Persian misspellings caused by a single omission. We also calculated the probability of occurrence and the distance cost of omission of one single letter in Section 5.1. However, as the misspellings caused by the omission of diacritics are a subset of omission misspellings, and given that words that need diacritics are usually (about 88% referring to Table 4) misspelled, we must consider the distance of omission of a single letter in a way that provides a proper support for diacritic misspellings too. Therefore, referring to (2) and given
that 88% of diacritics are omitted from words, we define the distance of omission of a diacritic as $1 - 0.88 = 0.12$. Algorithm 9 shows the distance of omission of one letter, including diacritics.

**Algorithm 9** Proposed Omission Distance of One Letters

```plaintext
proc OmissionDistance(c)
    if (c ∈ PersianDiacritics) /*{‘،’, ‘،’}*/
        then cost_o ← 0.120
    else cost_o ← Distance_e(omission) /*From Equation (2) and Table 5*/
return cost_o
```

5.4 The proposed string distance metric

The proposed character distance presented in the previous sections calculates only the distance of substituting a letter by another one considering the effects of keyboard layout on misspellings, the effects of upper characters on misspellings, homophone misspellings, and homomorph misspellings. Therefore, we calculate the distance of the rest of single-errors (insertion and omission of one letter and transposition of two adjacent letters) statistically as reported in Table 5. Algorithm 10 shows our proposed string distance metric, which supports single-error distribution patterns of language, the effects of keyboard layout on typographical errors, shift key and upper characters, homophone misspellings, homomorph misspellings, and the omission of diacritics. Function $f_k(i, j)$ is initially set to 0 for all $i$ and $j$, then $f_k(i, j)$ is calculated for all misspelling’s letters (misspelling word is indicated by $q$ and misspelling’s letters are indicated by $q_i$) and all replacement candidate letters (the replacement candidate is indicated by $l_j$). In each state, we increase the previous state distance with the minimum of the distance of transition from previous state to current state by each single-error type. Finally, $f_k(|q|, |l|)$ yields the total distance between the misspelling letter and

**Algorithm 10** Proposed String Distance Metric

```plaintext
proc KashefiStringDistance(q, l)
    for all i where j = 0 and for all j where i = 0 do $f_k(i, j) ← 0$
    $d(q, ε) ← OmissionDistance(l_i) /*From Algorithm 9*/$
    $d(ε, l_j) ← Distance_e(insertion) /*From Equation (2) and Table 5*/$
    $d(q_i, l_j) ← SubstitutionDistance(q_i, l_i) /*From Algorithm 8*/$
    if ($q_i = l_{i−1}$) and ($q_{i−1} = l_j$)
        then $t(q_i, l_j) ← Distance_e(transposition) /*From Equation (2) and Table 5*/$
    else $t(q_i, l_j) ← 2 /*Relatively Large Value*/$
    $f_k(i, j) ← \min(f_k(i−1, j) + d(q, ε), f_k(i, j−1) + d(ε, l_j), f_k(i−1, j−1) + d(q_i, l_j), f_k(i−2, j−2) + t(q_i, l_j))$
return $\frac{f_k(|q|, |l|)}{\max(|q|, |l|)}$
```
its replacement candidate. The returned distance is a normalized value from 0 to 1, where 0 represents the maximum similarity (sameness) between given strings and 1 represents the maximum distance (completely different).

6 Evaluation

In the previous sections, we presented a string distance metric customized for the Persian language that considered the error distribution pattern, keyboard layout, homophones, homomorphs, and diacritics in its calculations. In this section, we evaluate the effectiveness of this string distance metric through ranking respelling suggestions.

6.1 Methodology

To evaluate the proposed string distance metric, we have extracted and proof checked misspelled words from three large Persian corpora. We selected a list of misspelled words from the Persian corpora presented in Section 4. We also collected a dictionary of Persian words containing the correct spellings of all misspelled words. We excluded context sensitive misspellings from our list of misspelled words as being out of the scope of this paper.

To find the replacement suggestions, we generated a variation of every single-error on a misspelled word by $85n + 41$ transformations for a word with $n$ letters.\(^2\) We generated transforms in the edit distance of one and two with the misspelled word and then looked up transforms in the dictionary to extract the lexical word replacement suggestions. To enrich the transforms list, we also generated transforms by substituting misspelled word’s letters with their homophone and homomorph letters in all edit distances.

In about 98% of misspellings, transforms that were generated in the edit distance of one and two included at least one lexical word. Surprisingly, in the remaining 2% of misspellings, there was at least a lexical suggestion between homophone or homomorph transforms, which we generated in every possible distance. In other words, misspellings with more than two single errors were homophone or homomorph errors. Therefore, we never required to generate any transforms with more than two edit distances with misspellings in our benchmark. It must be noted that in about 20% of misspelling cases, lexical words that existed in the homophone transforms were overlapped with the lexical words that existed in the transforms with one or two edit distances from misspelling cases; this rate was about 35% for homomorph transforms.

The average length of a misspelling was six letters. On average for each misspelling, we generated 505 transforms with the edit distance of one, wherein four transforms

\(^2\) The number of letters in the Persian alphabet is thirty-two, but we have five more for Hamza, one more for pseudo-space (ZWNJ), and four more for diacritics. Therefore, a word can have $41n$ transforms by substituting one of its letters with another letter, $42(n+1)$ transforms by inserting one additional letter, $(n − 1)$ transforms by transposition of two of its adjacent letters, and $n$ transforms by deletion of one of its letters, where $n$ is the length of the misspelled word.
(less than 1%) were lexical words. Also in 11% of cases, no lexical word was found in transforms with the edit distance of one. For each misspelling, we also generated an average of 2,06,430 transforms with the edit distance of two, wherein six transforms (much less than 1%) were lexical words. We also generated 480 homophone transforms for each misspelling that had a homophone equivalent, wherein only one transform (about 0.2%) was lexical word. In addition, we generated thirty-four homomorph transforms for each misspelling that had a homomorph equivalent, wherein only one transform (about 3%) was lexical word. In rare cases, homophone and homomorph transforms contained more than one lexical word. This procedure for generating lexical words never failed; it always succeeded in generating the correct suggestion for every misspelling.

After extracting the lexical word suggestions for each misspelling, we ordered them using different string distance metrics and finally evaluated the effectiveness of each metric. In the context of spell checking and suggesting a list of respelling candidates as a ranked sequence of words for a misspelling, one cannot easily decide which suggestions must be included in the suggestion list (except the correct suggestion) and which suggestions must not be included in it; i.e. there is no true negative. Thus, the customary evaluation measures similar to traditional precision, recall, accuracy, F-measure, and fall-out measures cannot be used in this context. For example, some researchers (Brill and Moore 2000; Toutanova and Moore 2002; Hodge and Austin 2003; Naseem and Hussain 2007; Mitton 2009) have used the evaluation measure in (4). This evaluation measure gives the precision of correct suggestions ranked equal or below n.

\[ P_n = \frac{\text{Number of Times Rank}_{\text{Correct Suggestion}} \leq n}{|\text{Misspellings}|} \]  

Equation (4) does not present a unique measure for the evaluation of the total effectiveness of the ranking process; precision must be calculated separately for all desired places of correct suggestions in the list. Therefore, we have used the Mean Average Precision (MAP) (Keen 1971) and the Mean Reciprocal Rank (MRR) (Kantor and Voorhees 2000) as our evaluation measures, which are formulated by (5) and (6), respectively:

\[ \text{MAP} = \frac{1}{10} \sum_{n=1}^{10} P_n \]  
\[ \text{MRR} = \frac{1}{|\text{Misspellings}|} \sum \frac{1}{\text{Rank}_{\text{Correct Suggestion}}} \]  

Equation (5) differs a little bit from the main definition of MAP measure presented by Keen (1971) as all the retrieved suggestions relate to misspellings. The upper

---

3 The rank of an element \( e \) in the list \( l \), shown by \( \text{Rank}_{e} \), is the place of the element \( e \) in the list \( l \).
range of the calculated average precision is also limited to 10 because spell checkers usually do not show more than ten suggestions.

Taking into account that our string distance metric depends on the distribution pattern of single-errors that were analytically derived from the same corpora that were used as the test bed, we further used a five-fold cross-validation technique to ensure reporting fair, reliable, and significant results. Our corpus was randomly split into five parts (folds), wherein each fold contained one-fifth of the Hamshahri, CRCIS, and SCE-IUST corpora. The algorithm used the distribution pattern of the misspellings in four of the folds when ranking candidates for the misspellings in the fifth; in other words, we trained it on four-fifths of the data and tested it on the last fifth. This process was repeated for every five folds and evaluation measures for each fold were calculated. For the sake of brevity, we hereafter only report the arithmetic mean (average) of the evaluation measures.

6.2 Results

We presented the most related works on spell checking and ordering spelling suggestions in Section 2. Here we compare the effectiveness of our proposed string distance metric to the metrics used in these related works. We have studied three major related researches on the Persian and similar languages. Naseem and Hussain (2007) have presented an approach for ordering Urdu suggestions. Since they have used hardcoded and non-adaptive similarity keys for homophones and homomorphs with no comparative evaluation of their approach, we could not implement and evaluate their approach to compare with our approach. They have however reported the $P_1$ score of their proposed approach to be 71.7% on ranking the Urdu spelling suggestion. Shaalan et al. (2003) have presented a spell checker for Arabic. They did not however consider the ranking process, which is our focus here in the current paper.

In addition, as approaches using similarity and phonetic keys (Odell and Russell 1918; Zobel and Dart 1996; Holmes and McCabe 2002; Hodge and Austin 2003) have reported low precision, and probabilistic and statistical methods (Bledsoe and Browning 1959; Pollock and Zamora 1984) were infeasible due to the lack of resources in the Persian language, we were compelled to implement and evaluate only the string distance metrics. We also defined the costs of insertion, deletion, and substitution of one letter for the Wagner–Fischer metric based on what we statistically analyzed from our large Persian corpora (reported in Section 5.1 and Table 5).

In order to evaluate our proposed string distance metric more precisely and to study the effects of each part of our metric, we incrementally evaluated our proposed metric in five steps. In the first step, we evaluated how effective our proposed metric would be if we just consider single-errors edit distance cost, which we statistically analyzed from our large Persian corpora. In the second step, we evaluated our proposed metric by considering the effects of keyboard layout on spelling errors too. In the third step, we evaluated our metric augmented by special considerations for homophone misspellings. In the fourth step, we included diacritic misspellings
Table 8. Comparative evaluation measures of the proposed metric

<table>
<thead>
<tr>
<th>Metrics</th>
<th>$P_1$</th>
<th>$P_{10}$</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming</td>
<td>0.604</td>
<td>0.961</td>
<td>0.875</td>
<td>0.732</td>
</tr>
<tr>
<td>Jaro–Winkler</td>
<td>0.575</td>
<td>0.941</td>
<td>0.857</td>
<td>0.705</td>
</tr>
<tr>
<td>Levenshtein</td>
<td>0.630</td>
<td>0.964</td>
<td>0.890</td>
<td>0.753</td>
</tr>
<tr>
<td>Damerau–Levenshtein</td>
<td>0.630</td>
<td>0.964</td>
<td>0.889</td>
<td>0.752</td>
</tr>
<tr>
<td>Wagner–Fischer</td>
<td>0.604</td>
<td>0.970</td>
<td>0.914</td>
<td>0.824</td>
</tr>
<tr>
<td>Our metric just with statistics of misspellings</td>
<td>0.742</td>
<td>0.974</td>
<td>0.914</td>
<td>0.824</td>
</tr>
<tr>
<td>Our metric considering keyboard layout</td>
<td>0.826</td>
<td>0.989</td>
<td>0.942</td>
<td>0.871</td>
</tr>
<tr>
<td>Our metric considering homophones</td>
<td>0.848</td>
<td>0.997</td>
<td>0.961</td>
<td>0.903</td>
</tr>
<tr>
<td>Our metric considering diacritic</td>
<td>0.853</td>
<td>0.997</td>
<td>0.964</td>
<td>0.906</td>
</tr>
<tr>
<td><strong>Our final metric considering homomorphs</strong></td>
<td><strong>0.862</strong></td>
<td><strong>1.000</strong></td>
<td><strong>0.978</strong></td>
<td><strong>0.919</strong></td>
</tr>
</tbody>
</table>

and reevaluated the metric. Finally, we evaluated the effectiveness of the complete proposed metric by considering homomorph misspellings.

Table 8 presents the comparative evaluation measures of our proposed string distance metric versus other related metrics through the ranking of the Persian-spelling suggestions. $P_1$ represents the precision of the ranking of a correct suggestion as the first element in the spelling suggestions list. $P_{10}$ shows the precision when the correct suggestion is in the first ten suggestions in the suggestions list. The MAP column stands for the mean average precision of ranking the spelling suggestions for each metric and MRR stands for the mean reciprocal rank measure of each metric through the ranking of the spelling suggestions.

Figure 3 shows the relative effectiveness of our proposed string distance metric compared to other string distance metrics through scatter plots of $P_1$, $P_{10}$, MAP, and MRR evaluation measures. The evaluation measure score of our proposed metric is shown on the $y$-axis and the evaluation measure score of other metrics are shown on the $x$-axis in Figure 3. Points above the line $y = x$ indicate higher effectiveness of our proposed metric than other metrics.

The results reported in Table 8 and Figure 3 show the effectiveness of our proposed metric in ranking respelling suggestions for any type of misspellings altogether. Table 9 presents the detailed comparative evaluation measures of string distance metrics for different misspelling types, namely the homophone, homomorph, and diacritic misspellings.

### 6.3 Analysis

Referring to Table 8, the Hamming metric has the worst results of ranking respelling suggestion due to its simple algorithm wherein different types of typographical errors are considered the same. The Jaro–Winkler metric’s results are the second lowest. This is due to the way the Jaro–Winkler metric treats common prefixes at the start of misspelled words and suggested words, when a misspelled word and its suggested word do not have a common prefix string. For example, when an error appears at the prefix of the word, the similarity score is lower in the Jaro–Winkler metric than in other metrics. This might be because the spelling errors in initial letters or prefix
Table 9. Comparative evaluation measures of the proposed metric for different misspelling types

<table>
<thead>
<tr>
<th>Misspelling type</th>
<th>Metrics</th>
<th>( P_1 )</th>
<th>( P_{10} )</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homophones</td>
<td>Hamming</td>
<td>0.764</td>
<td>0.986</td>
<td>0.932</td>
<td>0.847</td>
</tr>
<tr>
<td></td>
<td>Jaro–Winkler</td>
<td>0.821</td>
<td>1.000</td>
<td>0.972</td>
<td>0.856</td>
</tr>
<tr>
<td></td>
<td>Levenshtein</td>
<td>0.818</td>
<td>1.000</td>
<td>0.975</td>
<td>0.898</td>
</tr>
<tr>
<td></td>
<td>Damerau–Levenshtein</td>
<td>0.764</td>
<td>0.986</td>
<td>0.931</td>
<td>0.846</td>
</tr>
<tr>
<td></td>
<td>Wagner–Fischer</td>
<td>0.781</td>
<td>0.986</td>
<td>0.936</td>
<td>0.856</td>
</tr>
<tr>
<td></td>
<td><strong>Our proposed metric</strong></td>
<td><strong>0.966</strong></td>
<td><strong>1.000</strong></td>
<td><strong>0.996</strong></td>
<td><strong>0.981</strong></td>
</tr>
<tr>
<td>Homomorphs</td>
<td>Hamming</td>
<td>0.621</td>
<td>0.979</td>
<td>0.872</td>
<td>0.737</td>
</tr>
<tr>
<td></td>
<td>Jaro–Winkler</td>
<td>0.638</td>
<td>0.972</td>
<td>0.885</td>
<td>0.761</td>
</tr>
<tr>
<td></td>
<td>Levenshtein</td>
<td>0.648</td>
<td>0.972</td>
<td>0.888</td>
<td>0.766</td>
</tr>
<tr>
<td></td>
<td>Damerau–Levenshtein</td>
<td>0.617</td>
<td>0.962</td>
<td>0.861</td>
<td>0.732</td>
</tr>
<tr>
<td></td>
<td>Wagner–Fischer</td>
<td>0.600</td>
<td>0.951</td>
<td>0.832</td>
<td>0.713</td>
</tr>
<tr>
<td></td>
<td><strong>Our proposed metric</strong></td>
<td><strong>0.934</strong></td>
<td><strong>1.000</strong></td>
<td><strong>0.990</strong></td>
<td><strong>0.962</strong></td>
</tr>
<tr>
<td>Diacritics</td>
<td>Hamming</td>
<td>0.758</td>
<td>1.000</td>
<td>0.975</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>Jaro–Winkler</td>
<td>0.758</td>
<td>1.000</td>
<td>0.975</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>Levenshtein</td>
<td>0.758</td>
<td>1.000</td>
<td>0.975</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>Damerau–Levenshtein</td>
<td>0.758</td>
<td>1.000</td>
<td>0.975</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>Wagner–Fischer</td>
<td>0.382</td>
<td>0.987</td>
<td>0.864</td>
<td>0.610</td>
</tr>
<tr>
<td></td>
<td><strong>Our proposed metric</strong></td>
<td><strong>1.000</strong></td>
<td><strong>1.000</strong></td>
<td><strong>1.000</strong></td>
<td><strong>1.000</strong></td>
</tr>
</tbody>
</table>

Fig. 3. Relative effectiveness of the proposed string distance metric compared to other metrics.
of words in Persian happens at the same rate of spelling errors in the middle or final sections of words as mentioned in Section 4.

The Levenshtein and Damerau–Levenshtein metrics have nearly similar effectiveness on ranking respelling suggestions. However, the Damerau–Levenshtein metric considers the cost of transposition of two adjacent letters as one edit distance, while the Levenshtein distance metric considers it as two edit distances (one deletion and one insertion). Considering that the Persian language has few transposition-type misspellings, as it was noted in Section 4, a significant difference does not show up in the Levenshtein and Damerau–Levenshtein evaluation results. The Wagner–Fischer metric customized to the Persian specific single-error costs (as described in Section 5.1) gave the best results among others except for our metric.

As shown in Table 8, our metric with just single-error edit distance cost statistics showed approximately similar results as the Wagner–Fischer metrics on ranking respelling suggestions and the slight differences were due to the fact that we have considered the transposition as a single-error whereas Wagner–Fischer has considered it as two single-errors. The effectiveness of the ranking process improved remarkably when we considered the effects of the keyboard layout and upper characters. Consideration of homophone misspellings improved the process of ranking suggestions too because the Persian language consists of many homophone letters that can lead to high rate of orthographical homophone misspellings. As diacritics constitute few language words that may be misspelled, their consideration did not improve the results remarkably. Special support for homomorphs did not present an improvement as much as homophones and keyboard layout did. The reason is that most of homomorph letters in the Persian keyboard layout are adjacent and the highest amount of similarity is calculated for homomorph words just by considering the keyboard layout effect.

As it is shown in Table 9, our metric preserved indisputably the best effectiveness on ranking homophones, homomorphs, and diacritic misspellings. Our proposed metric was on average about 20% more effective in ranking respelling suggestions of homophone misspellings, about 30% more effective for homomorphs, and about 35% more effective in ranking respelling suggestions of diacritic misspelling in contrast to other metrics in Persian language. The results of ranking diacritic misspellings looked strange. The results of the Wagner–Fischer metric were far out of line, while the results of the Hamming, Levenshtein, and Damerau–Levenshtein metrics yielded completely identical results. Diacritic misspellings differed from correct suggestions with the omission of a letter (i.e. a diacritic). Moreover, in most cases, a lexical suggestion that differed from a diacritic misspelling with the substitution of a letter already existed in the suggestions list. Given that we had configured the Wagner–Fischer metric with the Persian misspellings’ statistics, and given that the cost of substitution of two letters is lower than the omission of a letter according to Table 5, the Wagner–Fischer metric ranked suggestions that had a substitution distance higher than suggestions that had an omission distance. This resulted in missing the correct suggestion from the top of the list of suggestions ($P_1$). The Hamming, Levenshtein, and Damerau–Levenshtein metrics considered the same distance for omission and substitution but suggestions that had an omission distance,
including the correct suggestion, were one letter (i.e. a diacritic) longer than those that had a substitution distance. Because the Hamming, Levenshtein, and Damerau–Levenshtein metrics calculated a normalized score by dividing the distance by the length of the word, suggestions that had an omission distance, including the correct suggestion, were ranked higher than suggestions with substitution distance, resulting in better ranking of diacritic misspellings than the Wagner–Fischer metric.

7 Conclusion

In this paper we proposed a novel string distance metric to rank and select suitable suggestions from the list of respelling candidates of a misspelled word with a single misspelled letter or multiple misspelled letters. Our focus was on the Persian spell correction, so we discussed spelling aspects of the Persian language first. We also statistically analyzed the spelling errors caused by single letter error and its patterns, spelling errors caused by errors in multiple letters of a word, homomorph spelling errors, homophone spelling errors, and diacritic spelling errors. Based on these statistics, we adjusted the cost of each edit distance while preserving the effects of keyboard layout on typographical spelling errors such that closer letters on the keyboard layout were more probable to be mistaken than farther ones. In addition, as considerable amount of Persian letters are upper characters, we especially focused on upper characters.

We evaluated the rate of correct suggestions ranked as the top of the list ($P_1$), as well as the rate of correct suggestions ranked in the top ten places ($P_{10}$) in the list of suggestions. We also evaluated the MAP of ranking correct suggestions in the top ten places in the list of suggestions, and the MRR of the rank of correct suggestions in respelling candidates’ list for Persian misspellings extracted from three large Persian corpora. Evaluation results demonstrated the superiority of our proposed metric over the $P_1$, $P_{10}$, MAP, and MRR metrics in ranking respelling candidates of a misspelling. Evaluation results also showed substantial superiority of the effectiveness of our proposed metric in contrast to other metrics in ranking respelling suggestions of homophone, homomorph, and diacritic misspellings.

As a future work, we are interested to improve the effectiveness of the Persian spell checking approach by combining N-Gram models and the frequency of words with the proposed metric, especially for context-sensitive misspellings.

Acknowledgments

The work reported in this paper is a partial result of the industrial research project carried out in Iran University of Science and Technology (IUST) under contract (numbered 48-M-10330) with the Supreme Council of Information and Communication Technologies (SCICT), Tehran, Iran. We also thank our colleagues Mitra Nasri, Sina Iravaninan, and Mohammad Hedayati Gudarzi.

References


A metric for ranking Persian respelling suggestions


