On the Development of Name Search Techniques for Arabic

Syed Uzair Aqeel, Steve Beitzel, Eric Jensen, David Grossman, and Ophir Frieder
Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616. E-mail: {aqeel, beitzel, jensen, grossman, frieder}@ir.iit.edu

The need for effective identity matching systems has led to extensive research in the area of name search. For the most part, such work has been limited to English and other Latin-based languages. Consequently, algorithms such as Soundex and *n*-gram matching are of limited utility for languages such as Arabic, which has vastly different morphologic features that rely heavily on phonetic information. The dearth of work in this field is partly caused by the lack of standardized test data. Consequently, we have built a collection of 7,939 Arabic names, along with 50 training queries and 111 test queries. We use this collection to evaluate a variety of algorithms, including a derivative of Soundex tailored to Arabic (ASOUNDEX), measuring effectiveness by using standard information retrieval measures. Our results show an improvement of 70% over existing approaches.

Introduction

Identity matching systems frequently employ name search algorithms to locate relevant information about a given person effectively. Such systems are used for applications as diverse as tax fraud detection and immigration control. Using names to retrieve information makes such systems susceptible to problems arising from typographical errors. That is, exact match search approaches will not find instances of misspelled names or those names that have more than one accepted spelling. An example of the importance of quality identity matching is noted in an NCR (1998) report that estimates that the state of Texas saved $43 million over 18 months in the area of tax compliance by using an improved name search system. Thus, the importance of such name-based search applications has resulted in improved name matching algorithms for English that make use of phonetic information, but these language-dependent techniques have not been extended to Arabic.

Arabic retrieval has been studied since the late 1980s (Tayli & Al-Salamah, 1990; Al-Shalabi & Evens, 1998). Arabic name search in particular, however, to our knowledge has not been studied. Lack of a standard test data set has certainly been part of the problem. To address this deficiency, we have built a test collection of 7,939 Arabic names and 111 queries with corresponding relevance judgments. We employ both phonetic and string similarity-based name search algorithms. We developed an Arabic counterpart to English Soundex, ASOUNDEX, measuring the effect of Arabic-specific modifications at each step. We also tested language-independent *n*-grams and string distance techniques on our Arabic collection. Finally, we experimented with combination of evidence, fusing ASOUNDEX with *n*-grams and edit distances to improve effectiveness. Our results show that our best fusion technique provides an improvement of more than 70% over basic *n*-gram techniques.

Received July 28, 2003; revised February 21, 2005; accepted March 9, 2005

 • Published online 21 February 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.20323

Previous Work

Numerous name search algorithms for Latin-based languages effectively find relevant identification information. Algorithms that use the phonetic features of names have been researched thoroughly for English, and string similarity techniques have garnered interest because of their language-independent methodology. In proposing a method for automatic correction of spelling errors, Damerau (1964) identified four major categories. These are shown in Table 1, which includes examples in both English and Arabic. Although Damerau's research was limited in scope to English, the categories identified are applicable to text in other languages, including Arabic.

Phonetic Matching Algorithms

The United States Census Bureau designed Soundex in 1935. Binstock and Rex (1995) describe it as a hashing algorithm that encodes words into a fixed-length Soundex code. The algorithm partitions consonant sounds into categories based on distinctive phonetic features, as shown in Table 2.

To encode a name, the algorithm retains the first character of the name, and then replaces succeeding consonants

TABLE 1. Common spelling errors.

	English		Arabic	
Type of error	Baseline name	Deviation	Baseline name	Deviation
Insertion	Fisher	Fischer	حميد	حميييد
Omission	Johnston	Johnson	احمد	امد
Substitution	Catherine	Katherine	صابر	سابر
Transposition	Hagler	Halger	وجاهت	وجهات

TABLE 2. English Soundex codes and categories.

Soundex code	Characters	Category
1	*b*, *f*, *p*, *v*	Labial
2	*c*, *g*, *j*, *k*, *q*, *s*, *x*, *z*	Guttural and sibilants
3	*d*, *t*	Dental
4	*l*	Long liquid
5	*m*, *n*	Nasal
6	*r*	Short liquid

with the numeric values shown in Table 2. Vowels and the characters *h*, *w*, and *y* are ignored because the phonetic information they provide is often ambiguous when they are combined with other characters. Adjacent repeated consonant sounds are represented only once in the code, introducing resilience against inadvertent insertions and deletions in names. Conflating closely linked consonant sounds such as *c* and *s* allows for toleration of common spelling mistakes, but as Gadd (1988) states, the categorization is often too general, as in associating *k* and *s*.

Gadd's (1990) PHONIX is a more discriminating phonetic algorithm that performs substitutions on about 160 character combinations before performing Soundex-style encoding. The letter *x*, for example, is replaced by *ecs* before the codes shown in Table 3 are applied.

Combined with the substitutions performed in the preprocessing phase, the expanded PHONIX code-set partitions phonetic sounds more effectively, with *c* and *s* differentiated by the definition of an entirely different category, which reduces the impact of substitution errors. In evaluating PHONIX for the general English name search task, Hodge and Austin (2001) note that its substitution rules were originally designed for South African name matching, and that this feature limits its applicability. This limitation may be circumvented by the design and inclusion of additional substitution rules, as done by Rogers and Willett (1991).

TABLE 3. English PHONIX codes.

PHONIX code	Characters
0	*a*, *e*, *i*, *o*, *u*, *h*, *w*
1	*b*, *p*
2	*c*, *g*, *j*, *k*, *q*
3	*d*, *t*
4	*l*
5	*m*, *n*
6	*r*
7	*f*, *v*
8	*s*, *x*, *z*

The codes generated by the original Soundex algorithm are always four letters in length. This length is adequate for most purposes but can result in the truncation of potentially significant phonetic information when the name being encoded has more than three consonant sounds. In evaluating the PHONIX algorithm, Zobel and Dart (1996) also considered a variant that did not place a limit on the length of the code generated. Experimentation revealed that this variant outperformed the original algorithm.

In adapting the Soundex system for the Thai language, Suwanvisat and Prasitjutrakul (1998) showed that the algorithm's principles could be used for phonetic matching in other languages as well. Similarly, Kang and Choi (2001) partitioned the Korean alphabet in developing their Kodex algorithm. Echoing the findings of Zobel and Dart (1996), Suwanvisat and Prasitjutrakul (1998) found that effectiveness was considerably improved by removing the limit on code length.

Recent work on improving Soundex focuses primarily on improving performance by manipulating names before encoding or by altering Soundex codes after encoding. Celko's (1995) approach, for example, ignores *h* only when it is not preceded by an *a*, rather than doing so indiscriminately. Holmes and McCabe (2002), conversely, describe a feature they call *code shifting*, in which the second character of the Soundex code is removed. They show this feature has a significant impact in dealing with errors of insertion and omission near the beginning of names. Significant deviations from Soundex have included Zobel and Dart's (1996) Editex and Hodge and Austin's (2001) Phonetex. Unlike Soundex and PHONIX, the former does not require that characters be placed in a single category, allowing for a more natural representation of the phonetic similarities between characters. Thus, Editex allows the characters *c* and *s* to exist in multiple categories to reflect their relationships with other phonetically similar characters. Hodge's Phonetex is a pared-down implementation of PHONIX featuring nonspecific substitution rules and an expanded set of disjoint codes, which reduces the granularity of the retrieval system, improving effectiveness and making it more suitable for general name search applications than the original PHONIX algorithm.

String Similarity Measures

An alternative approach is the use of string distance measures and *n*-grams. An *n*-gram is a set of character sequences of length *n* extracted from a word. The *n*-gram techniques are language independent, differing significantly from Soundex in that they do not rely on phonetic similarity. As identified by *n*-grams, similarity is based purely on spelling rather than phonetic information (Hall & Dowling, 1980; Pfeifer, Poersch, & Fuhr, 1995). The two most commonly used values of *n* are 2 and 3 (bigrams and trigrams). A value of 1 results in a character-by-character matching mechanism. Using a value larger than 3 often yields similar results

DOI: 10.1002/asi

TABLE 4. *N*-grams yield by "Davidson."

Type	*n*	Grams
Bigrams	2	'Da,' 'av,' 'vi,' 'id,' 'ds,' 'so,' 'on'
Trigrams	3	'Dav,' 'avi,' 'vid,' 'ids,' 'dso,' 'son'
Padded bigrams	2	'_D,' 'Da,' 'av,' 'vi,' 'id,' 'ds,' 'so,' 'on,' 'n_'
Padded trigrams	3	'_Da,' 'Dav,' 'avi,' 'vid,' 'ids,' 'dso,' 'son,' 'on_'

because names less than or equal to 4 characters in length will be compared directly. Experimentally, Pollock and Zamora (1981) have shown that bigrams and trigrams generally outperform other *n*-grams. An important variant of standard *n*-grams are "padded" *n*-grams: the query name has spaces appended before and after it. Pfeifer and associates (1995, 1996) determined this feature improves effectiveness by increasing the importance of matching the initial and terminal letters of a word. Table 4 shows plain and padded grams for the name *Davidson* with *n* at 2 and 3.

Zamora and colleagues (1981) and Angell and coworkers (1983) considered the use of trigrams to conflate relevant words that have been spelled differently. Introducing a similarity measure, Damerau (1964) allowed for ranking of retrieved words by counting differences in features extracted from the word and the query. Pfeifer and associates (1995) also proposed a similarity measure, which measured similarity by counting the number of *n*-grams common to two words. Holmes and McCabe (2002) showed that for English, *n*-gram techniques are less effective than Soundex-based techniques. The explanation provided is that because *n*-grams are unaware of the phonetic information Soundex uses, they are not able to recognize the phonetic equivalence of various characters. Even so, Hodge and Austin (2001) note that *n*-gram techniques are better equipped to handle insertion and deletion errors.

An alternative string similarity measure to *n*-grams are edit distances. The most common edit-distance measure, the Levenshtein distance, represents the smallest number of character insertions, deletions, and substitutions required to transform one string into another (Levenshtein, 1965). Thus, two names that differ only by a single character have an edit distance of 1. Zobel and Dart (1995, 1996) investigated edit distances as a stand-alone approximate search tool and found that they outperform phonetic matching techniques. Edit distances are also an integral element of Zobel's Editex algorithm, in which they are used to determine the similarity of words after phonetic encoding has been completed. It is significant to note that, unlike other ranking measures, edit distances are not calculated in linear time; given two names, of length *n* and *m*, their edit distance would be computed in $\Theta(nm)$. Also, edit distances must be computed at run time, but other techniques such as Soundex and *n*-grams allow retrieval systems to store encoded names and simply use them at run time. As Erikson (1997) notes, the high cost of doing this at run time makes database partitioning essential to maintain reasonable efficiency in practical systems.

Evaluation

Name search may be evaluated similarly to information retrieval (IR) tasks, thereby allowing us to use standard precision-and-recall-based measures, as demonstrated by Zobel and Dart (1996) and Holmes and McCabe (2002). The latter also uses average precision, which interpolates precision over 11 points of recall from 0% to100%. *R*-precision is also useful, because the number of relevant results in name search is often too small for accurate interpolation.

To apply precision and recall, it is necessary for retrieval techniques to return ranked results. Pfeifer's similarity measure makes this process easily possible for *n*-grams, but the original Soundex algorithm is limited to a simplistic evaluation of results as either relevant or nonrelevant. Holmes circumvented this issue by employing the Dice (Dice, 1945) coefficient to rank results. As shown in Equation 1, this method measures the proportion of features shared by two sets of data (in this case, Soundex codes).

$$\delta = \frac{2(\chi)}{(\alpha + \beta)} \quad \text{Dice Co-efficient} \tag{1}$$

where δ is the similarity score, χ are the features common to both names, and α and β are the number of features of the first and second name, respectively. In applying Dice to Soundex, Holmes and McCabe (2002) considered each individual letter of a Soundex code to be a "feature." For *n*-grams, on the other hand, "features" might be each individual *n*-gram of the name.

Because the number of features is relatively small, results are often weakly ordered; several results have the same similarity score, and therefore equal rank. Raghavan and colleagues (1989) noted that precision is an unreliable measure when computed against a weakly ordered result set and proposed two alternatives that take a probabilistic approach. Seeking to employ standard measures for evaluation, Zobel instead shuffles results of equal rank to generate random permutations of the result set and then averages the precision figures for 10 such permutations.

Determining a definition for *relevant* in the context of name search poses another problem for applying information retrieval measures to name search. There have been several studies of the problem of determining the meaning of relevance (Borlund, 2003; Mizzaro, 1998). Zobel and Dart (1996), whose study was based on relevance as determined by phonetic similarity, instructed assessors to "regard a name and a query as a match if you think they may be a match, that is, a name and query are a match whenever you cannot be sure they are distinct."

Fusion

Combination of evidence, or data fusion, attempts to improve the effectiveness of retrieval systems by combining the results of a variety of retrieval methods. Fox and Shaw (1994) identified several algorithms for linearly combining

DOI: 10.1002/asi

the results of multiple retrieval techniques. These include COMBSUM, which simply sums similarity scores, and COMBAVG, which finds the arithmetic mean of scores from multiple techniques (McCabe, Chowdhury, Grossman, & Frieder, 1999). Both Holmes and McCabe (2002) and Zobel and Dart (1996) utilize fusion to improve the effectiveness of their name search systems.

Zobel and Dart's (1996) experiments, which fused various phonetic techniques with string similarity measures, determined that such combinations almost always provide superior performance. Holmes and McCabe (2002) similarly combined various iterations of Soundex along with *n*-grams to achieve higher levels of recall. In evaluating the results yielded by Phonetex, Hodge and Austin (2001) also recommend the integration of a typographic evaluation technique to introduce resilience against insertion and deletion errors.

Arabic Background

Arabic is one of the world's major languages and is spoken in a wide belt of nations extending from the Arabian Peninsula across North Africa to the Atlantic Ocean (Katzner, 2002). Although several distinct dialects thrive in the various Arabic-speaking countries, the written form of the language, Standard Arabic, is consistent and used in books and newspapers throughout the Arabic world. Unlike text in Latin-based languages, Arabic text is oriented right to left.

The Arabic alphabet consists of 29 letters and five basic diacritical marks. Diacritics are marks that attribute special phonetic values to certain characters, such as the cedilla of *façade*. This basic character set may be extended to 90 with the addition of other marks and vowels, as Tayli and Al-Salamah (1990) note. Further, because Arabic characters change shape when combined with other characters, the alphabet requires at least 102 glyphs for a basic representation (AbiFares, 1998). Coupled with the fact that the Arabic alphabet contains several sets of homophones, standardizing spellings for Arabic names in information systems is an error-prone process.

Two forms of vowels exist in Arabic. Long vowels are communicated by means of full characters, such as ا (*alif*), و (*waaw*), and ي (*yaa*). The diacritical symbols *fathah*, *dammah*, and *kasra* represent short vowel sounds. When two of these short vowels appear adjacently in text, they are referred to as *doubled diacritics* and are replaced by a single symbol referred to as *tanween*, which appends both a consonant and a vowel sound to the word. The sound of a consonant may be doubled by following it immediately with another diacritical symbol known as *shadda.* Although not a vowel, *shadda* is most often grouped with short vowels for symmetry. Quite often, short vowel sounds are not inserted fully in text, and the reader is expected to deduce their presence from contextual information. Consequently, Arabic words may be interpreted in as many as 15 different ways when considered individually, indicating an extremely complex morphological structure. Hebrew, considered the most complex language of the Semitic family, is by comparison relatively simple, featuring a maximum of eight interpretations for individual words (UBAccess, 2002).

Although Standard Arabic is consistent across the Arab world, it may still be divided into three distinct but commonly occurring categories: text that fully utilizes diacritical symbols to convey pronunciation information; text that makes use of diacritics, but not exhaustively; and text that contains no diacritics at all. The existence of these three categories, distinguished only by the presence of diacritics, adds complexity to the design of Arabic information systems: Although the phonetic information diacritics encapsulate is certainly valuable, it must be given only measured importance because diacritics, by definition, are less significant than consonants and vowels and are, in fact, irrelevant when unaccompanied. Al-Shalabi and Evens (1998) note that because computers represent diacritics as full characters, the distinction is lost, and incorrect usage can mislead information systems into marking relevant results as being irrelevant. Table 5 shows how the name *Mohammed*, spelled with irregular use of diacritics, may be represented in three completely different ways in extended ASCII encoding.

Consequently, improper diacritic usage may be considered a major category of errors in the same vein as those identified by Damerau (1964). Previous efforts to develop effective Arabic information systems have tackled the problem diacritics pose by removing them before performing any form of retrieval. In developing Arabic stemmers for unstructured text search systems, both Aljlayl and Frieder (2002) and Larkey, Ballesteros, amd Connell (2002) removed diacritics in a preprocessing phase. The fact that Arabic script may or may not clearly define word boundaries is another issue that requires appropriate consideration. Even when text is standardized, the possibility of incorrect insertion of spaces remains a valid concern. Wegener (2000) points out that a final concern lies in the fact that there are almost 20 encodings currently in use for Arabic, making it necessary for practical retrieval systems to be usable with all of them, and that requirement leads to a confusion that, again, commonly results in typographical errors.

Arabic names are different from English names in that they are meaningful words chosen from the language. The name صابر, for example, is the Arabic word for *patient*. This property has interesting implications for our research: Because names are meaningful words, their spellings are largely standardized, rendering one of the key motivations for English name search, multiple accepted spellings (as with *Catherine* and *Katherine*), inapplicable. Indeed, our primary motivation is the fact that irregular diacritic use and regional variants make typographical errors very common.

TABLE 5. Various representation of *Mohammed* based on diacritic usage.

Arabic name	ASCII representation
محمد	ÅÍâÏ
مُحَمَّد	ÅïÍîâÏ
محمّد	ÅÍâñÏ

DOI: 10.1002/asi

TABLE 6. Initial ASOUNDEX code set.

Code	Characters	English phonetic equivalent	Category
1	ب,ف	*b*, *f*	Labial
2	ك,ق,ظ,ص,س,ز,ج,خ	*k*, *q*, *z*, *s*, *c*, *z*, *j*, *kh*	Guttural and sibilants
3	ط,ض,ذ,د,ث,ت	*t*, *d*	Dental
4	ل	*l*	Long liquid
5	م,ن	*m*, *n*	Nasal
6	ر	*r*	Short liquid

Methodology

We developed two new name matching algorithms, ASOUNDEX and tanween-aware *n*-grams, and evaluated them against existing *n*-grams and edit distance techniques. As we are unaware of existing Arabic phonetic matching techniques, we only compare ours to these language-independent techniques. The algorithms experimented with are briefly outlined in the following:

I. Previous Work
   1. DiceUnordered similarity of *n*-grams including diacritics:
      a. Bigrams: *2*-grams
      b. Trigrams: *3*-grams
      c. Padded bigrams: *2*-grams including spaces at beginning and end of name
      d. Padded trigrams: *3*-grams including spaces at beginning and end of name
   2. Edit distances: Levenshtein distances representing the number of insertions, deletions, and substitutions

II. Our Algorithms
   1. DiceUnordered similarity of *n*-grams, which ignore all diacritics other than the emphatic *tanween* and *shadda*:
      a. *Tanween*-aware bigrams: *2*-grams
      b. *Tanween*-aware trigrams: *3*-grams
      c. *Tanween*-aware padded bigrams: *2*-grams including spaces at beginning and end of name
      d. *Tanween*-aware padded trigrams: *3*-grams including spaces at beginning and end of name
   2. ASOUNDEX-8, ASOUNDEX-9, ASOUNDEX-10, and ASOUNDEX-FINAL: Intermediate algorithms based on English Soundex, but augmented with Arabic-specific code categories.
   3. ASOUNDEX: Our final Arabic variant of the Soundex algorithm, which fuses multiple ASOUNDEX-FINAL codes of lengths between 2 and 7

ASOUNDEX

In designing our phonetic matching technique for Arabic, ASOUNDEX, we started with the approach used in Soundex of conflating similar-sounding consonants. We chose to retain the first character of the name being encoded, with consonants that followed being encoded according to our customized code set. This approach is similar to that employed in formulating Kodex (Kang & Choi, 2001). Following the example of Suwanvisat and Prasitjutrakul (1998) in Thai Soundex, we decided not to restrict the length of encoded words to 4, as in the original Soundex algorithm. Instead, we experimented with a variety of lengths to achieve optimal effectiveness. In Table 6, we illustrate the code set initially developed for ASOUNDEX.

These codes follow the phonetic categories declared by the original Soundex code set and capture the majority of Arabic consonants. The emphatic consonants ش and غ, however, have no counterparts in English and therefore require the definition of additional categories. In Table 7, we present the additions made to the original code set to reflect this. We refer to this code set as ASOUNDEX-8. Notice that although it may be argued that the consonant خ, similar to the *ch* in *Bach*, should similarly be given its own separate category, phonetics defines it as a member of the guttural group. We agree with this classification because it is, in fact, often mistaken for other members of the guttural group.

We also investigated the effects of categorizing the Arabic equivalents of the aspirate *h*. English Soundex and English PHONIX both specifically ignore *h* because its phonetic properties are not very pronounced and change in combinational forms such as *sh* and *ch*. This, however, is not true for Arabic, allowing us to conflate these characters by adding another category to the code set, creating a PHONIX-style preprocessing stage that normalizes the various Arabic aspirates. This extension to the ASOUNDEX-8 code set, shown in Table 8, is designated ASOUNDEX-9.

TABLE 7. Arabic-specific additions to the code set.

Code	Characters	English phonetic equivalent	Category
7	ش	*Sh*	Sharp dental
8	غ	*Gh*	Guttural aspirate

TABLE 8. Adding aspirates and vowels to the code set.

Code	Characters	English phonetic equivalent	Category
9	ة,ح,ه	*H*	Aspirate
A	و,ؤ	*W*	Labial semivowel
B	ع,ا,آ,ء,إ,أ	(Depends on diacritic used)	Vowel

DOI: 10.1002/asi

Also shown in Table 8 are two additional categories formed by conflating various long vowels. This, again, is a departure from the original Soundex algorithm, which ignores vowels completely. PHONIX performs similar conflation by indiscriminately replacing vowels with *V*. Because vowels are more clearly defined in Arabic and are not generally affected by the presence of other characters, they are more suitable for categorization than their English equivalents. We refer to these new variations as ASOUNDEX-10 and ASOUNDEX-FINAL.

The algorithm we refer to as ASOUNDEX uses the ASOUNDEX-FINAL encoding scheme to generate multiple codes, of lengths between 2 and 9, and then employs fusion to generate the best possible results using all these codes.

Tanween-Aware n-Grams

To compensate for improper use of diacritics, we considered an Arabic-specific variant of *n*-grams that ignores all diacritics other than the emphatic *tanween*, or doubled diacritics, and *shadda*. These diacritics are more significant than other diacritics such as *fathah* and *kasrah* because they append consonant sounds to words rather than simply altering vowel sounds. Table 9 shows how the presence of these emphatic diacritics alters pronunciation, with the appended consonant sounds highlighted. These *tanween*-aware *n*-grams allow for a fair comparison of *n*-grams and ASOUNDEX, which uniformly ignores diacritics, thereby effectively circumventing the problems of improper and incomplete usage of diacritics.

Similarity Coefficient

To rank names by similarity from the ASOUNDEX and *n*-gram algorithms (edit distances provide an inherent similarity measure), we employed the Dice coefficient used in previous work. We experimented with two versions of Dice: the first, which we refer to as *DiceExact*, requires that matching features occur at the same position within both names. The *n*-gram methods, however, are defined such that *n*-grams simply shared between both words are counted without regard to position. For a fair comparison, therefore, we also used DiceUnordered, which considers matching features as any features that are found within both names, regardless of position. When testing our various retrieval methods, we used the similarity measure most beneficial to effectiveness for each technique. Thus, we used DiceUnordered for *n*-gram techniques and DiceExact for ASOUNDEX. Because the number of features available for comparison is small, these measures often assign several names the same similarity score, leading to weakly ordered result sets. We follow Zobel's example in generating 10 random permutations of the result set returned and averaging precision values over these permutations to prevent this problem.

Results

Because no standard collection of Arabic names exists, we built one to test our algorithms. We started by pooling lists of names found on publicly available Web sites (e.g., kabalarians.com, arabia.com, and ajeeb.com). These lists included names that had been transliterated from Arabic to English, which we, therefore, traced back to their original Arabic spellings and then entered manually. We made every effort to ensure proper usage of diacritics. Importantly, the collection does not contain names prefaced with prefixes such as *Abu* or *Ibn*, which reflect familial relations. The name *Muhammed bin Sulaiman*, for example, actually translates as *Muhammed, son of Sulaiman*, so that *Muhammed* is the given name, and *Sulaiman* is the surname. Both the constituent names are stored in the collection, but the combined form is not stored, because storing it would be equivalent to storing both the given name and the surname in an English name database. It is assumed that words that express familial relations in this manner would be removed at index time. Instead, we tested retrieval effectiveness of compound names by formulating queries such as *Abdul Qadir*, which is a single name written as two distinct words. It compounds the words *Abdul* and *Qadir* and is therefore relevant to both.

Queries were formulated by applying Damerau's (1964) common spelling errors to randomly selected names from the collection. Thus, queries featured insertion, deletion, substitution, and transposition of both characters and diacritics. In addition, we applied random spacing to some of the queries to simulate word-break errors, which can occur because Arabic text does not always have clearly defined word boundaries. We note the difference from Zobel's method: Zobel used names from the collection as queries and then evaluated the results for relevance, a method invalid in our case because Arabic names are meaningful words (*Abdullah* means "servant of God") and therefore have only one accepted spelling. Because the collection, by definition, contains only accepted spellings, it is unsuitable as a source of queries. Consequently, we are forced to create queries by transforming names from the collection even

TABLE 9. The phonetic effects of *tanween* and *shadda*.

Diacritic	Symbol	Application	Baseline word	Deviation
Shadda	ّ	فكّر	*Fukr*	Fu-**kk**-ir
Double Fathah *(tanween)*	ً	مبكرً	*Mubakkir*	Mubakkir-**an**
Double Dammah (*tanween*)	ٌ	بنتٌ	*Bint*	Bint-**un**
Double Kasrah (*tanween*)	ٍ	واحدٍ	*Wahid*	Wahid-**in**

DOI: 10.1002/asi

though this process causes a natural bias in favor of string similarity measures, particularly edit distances.

An initial set of 50 training queries was prepared to tune parameters such as categorization of vowel sounds and optimal code length for the ASOUNDEX algorithm. A further set of 111 queries was then used for evaluation of the various algorithms being tested. Finally, a derivative set of queries was formulated by simply stripping all diacritics from this set of evaluation queries.

Before creating relevance judgments for the queries, we first defined a measure of relevance. Zobel and Dart's (1996) definition of relevance instructed assessors that "a name and a query are a match whenever you cannot be sure they are distinct," with words being read aloud to the assessors. We considered Mizzaro's (1998) review of definitions of relevance employed for information retrieval before using the following: *A name is relevant to a query name if they refer to the same name.* Leaning toward Mizzaro's "user-oriented" paradigm allows us to consider derivative forms of names as being relevant. In English, this would make *Robert* relevant to a query of *Rob*, *Bob*, or *Robert*; in Arabic, this includes partial matches for combinational forms such as *Abdul Qadir*.

The relevance judgments were made by a manual inspection of pooled results by an Arabic speaker on the basis of a manual inspection of the names rather than Zobel's pronunciation-based method. Zobel notes that our method would cause a bias in favor of string-distance measures such as *n*-grams and edit distances. Each query had, on average, 1.81 relevant names, with a standard deviation of 1.1.

ASOUNDEX Variants and Code Lengths

As described previously, we considered various code sets and several different code lengths in developing ASOUNDEX. In Table 10, we show the effectiveness of the variants we tested. In testing these intermediate versions of ASOUNDEX on our training set of 50 queries, we used a fixed code length of 4. The first character of each name was retained, and the next three consonants were encoded as previously discussed. Our first version of the algorithm, ASOUNDEX-8, is the most basic version that captures all the consonant sounds of Arabic. The results showed that conflating Arabic's various aspirates, as done by ASOUNDEX-9, provides a clear improvement in effectiveness. Similarly, grouping *waaw*'s, as done by ASOUNDEX-10, also provides a significant improvement. In light of this, we experimented with conflating other long vowels in a similar manner, producing ASOUNDEX-FINAL. Interestingly, this version did not provide performance superior to that of ASOUNDEX-10.

TABLE 10. ASOUNDEX variants at code length 4.

Algorithm	Average precision	*R*-precision
ASOUNDEX-8	0.2427	0.1285
ASOUNDEX-9	0.3005	0.2222
ASOUNDEX-10	**0.3452**	**0.2569**
ASOUNDEX-FINAL	0.3411	0.2344

TABLE 11. Fusing ASOUNDEX-10 and ASOUNDEX-FINAL codea.

Algorithm	Average precision	*R*-precision
Fused ASOUNDEX-10	0.4021	0.3236
Fused ASOUNDEX-FINAL	0.4495	0.3479

As reflected in Table 11, this experimentation shows that although it does not perform as well on its own at code length 4, ASOUNDEX-FINAL is a better candidate for fusion than ASOUNDEX-10. Consequently, our final algorithm, ASOUNDEX, fuses multiple ASOUNDEX-FINAL codes of different lengths.

Next, we examine the ways different code lengths and fused combinations of different code lengths perform. Figure 1 shows the ways altering the length of ASOUNDEX-FINAL codes between 2 and 9 affected average precision.

Retrieval effectiveness of ASOUNDEX (fused ASOUNDEX-FINAL) clearly improves as code length is increased; performance levels off as length approaches 6. We also show average precision for a single ASOUNDEX-FINAL code as length is increased. Again, there are significant improvements as length increases, and, as before, these gains level off as code length approaches 6. The finding that both fused and single codes perform similarly when code length is less than 3 is attributable to the fact that Arabic words are normally composed of only three or four consonants (Aljlayl & Frieder, 2002; Tayli & Al-Salamah, 1990). Therefore, we set the maximal code length of codes fused by ASOUNDEX to 7.

Comparison to String Matching Baselines

We then proceeded to use our testing set of 111 queries to compare ASOUNDEX with different candidate algorithms. Because we were testing for performance both with and without diacritics, we initially collected two sets of data. The first of these used queries with diacritics on our original collection, which also contained diacritics. The second was of queries without diacritics on a version of the collection

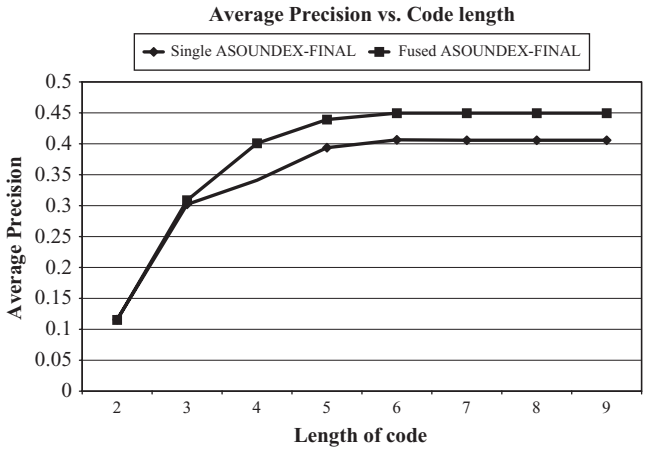

FIG. 1. Change in average precision with code length.

DOI: 10.1002/asi

TABLE 12. Effectiveness for queries with diacritics.

Algorithm	Average precision	*R*-precision	P @ 1	P @ 2	P @ 3
Edit distances	**0.5141**	**0.4356**	**0.3798**	**0.4494**	**0.4372**
Tanween-aware bigrams	0.4538	0.4173	0.3745	0.4089	0.4180
Tanween-aware padded bigrams	0.4789	0.4119	0.3768	0.4058	0.4196
Tanween-aware padded trigrams	0.4327	0.3737	0.3194	0.3768	0.3813
ASOUNDEX	0.4435	0.3576	0.2794	0.3573	0.3489
Tanween-aware trigrams	0.3533	0.3018	0.2488	0.3031	0.3026
Padded bigrams	0.3520	0.2595	0.2274	0.2740	0.2572
Padded trigrams	0.3164	0.2518	0.2503	0.2541	0.2572
Bigrams	0.2779	0.2335	0.1907	0.2213	0.2312
Trigrams	0.2535	0.2037	0.1884	0.2037	0.2182

TABLE 13. Effectiveness for queries without diacritics when used on a collection without diacritics.

Algorithm	Average precision	*R*-precision	P @ 1	P @ 2	P @ 3
ASOUNDEX	**0.5459**	**0.3787**	**0.2908**	**0.3596**	**0.3971**
Edit distances	0.4436	0.3125	0.2771	0.3474	0.3691
Padded bigrams	0.4411	0.2939	0.2343	0.3000	0.2962
Tanween-aware padded bigrams	0.4410	0.2939	0.2434	0.3084	0.2962
Padded trigrams	0.3844	0.2633	0.2457	0.2771	0.2794
Tanween-aware padded trigrams	0.3843	0.2633	0.2419	0.2801	0.2732
Bigrams	0.3228	0.2266	0.2037	0.2327	0.2312
Tanween-aware bigrams	0.3235	0.2266	0.1838	0.2266	0.2251
Trigrams	0.3024	0.2037	0.2182	0.2220	0.2083
Tanween-aware trigrams	0.3024	0.2037	0.2044	0.2106	0.2182

without diacritics. Because improper use of diacritics is a significant source of errors, however, we also constructed a third set of data that used queries without diacritics on the original collection that contained diacritics. In this way, we were able to test each algorithm's resilience to improper diacritic usage. Because there are, on average, only 1.81 relevant names for each query, we collected precision at one, two, and three names retrieved in addition to average precision and *R*-precision. Table 12 shows results from our first set of experiments, in which we ran queries with diacritics on the original collection. It is clear that ASOUNDEX represents a significant improvement over the most basic *n*-gram methods, but that improved string matching techniques such as *tanween*-aware padded bigrams provide superior performance. Edit distances perform best, with both high precision and high recall.

Welch *t* tests, however, show that the difference in effectiveness between ASOUNDEX and both edit distances and *tanween*-aware bigrams is insignificant at a confidence interval of 95%, with *p*-values of 0.1587 and 0.853, respectively.

In our second set of experiments, shown in Table 13, we stripped both the collection and the queries of diacritics and then measured performance. Although ASOUNDEX continues to perform well, effectiveness drops almost uniformly across the other techniques. We note that ASOUNDEX's *R*-precision remains stable, but that its average precision is markedly improved, indicating that relevant results are being attributed low rankings. This underlines our previous observation that average precision's interpolated process limits its applicability for name search.

Interestingly, the performance of edit distances is impacted drastically, even though with diacritics stripped from both the queries and collection, we would have expected it to remain stable. The *n*-gram techniques clearly benefit now that the ambiguity introduced by diacritics is completely removed, and they display markedly better performance across the board. This strips away the advantage that the *tanween*-aware techniques previously bore, and consequently *tanween*-aware techniques perform only as well as their regular counterparts.

Welch *t* tests show that the observed performance difference of ASOUNDEX over the best *n*-gram method, padded bigrams, is significant at a confidence interval of 95%, with a *p*-value of 0.0141. Comparing ASOUNDEX to edit distances, however, yields a *p*-value of 0.0818, indicating significance only at a 90% confidence interval.

To measure resilience to incorrect diacritic use, our third set of experiments paired queries stripped of diacritics with the original collection, which included diacritics. As we show in Table 14, ASOUNDEX's retrieval effectiveness for queries without diacritics is identical to that for queries with diacritics. This is simply because ASOUNDEX recognizes diacritics as short vowels and ignores them so that, in effect, all queries are stripped of diacritics. The effect on *n*-gram methods of stripping diacritics, on the other hand, is considerable. Padded bigrams and padded trigrams, previously the poorest of the *n*-gram methods, are now second only to ASOUNDEX. *Tanween*-aware *n*-grams, which rely on diacritics for additional phonetic information, are severely impacted by their absence, with extremely poor retrieval

DOI: 10.1002/asi

TABLE 14. Effectiveness for queries without diacritics when used on a collection with diacritics.

Algorithm	Average precision	*R*-precision	P @ 1	P @ 2	P @ 3
ASOUNDEX	**0.4435**	**0.3572**	**0.2972**	**0.3546**	**0.3627**
Padded bigrams	0.3303	0.2365	0.2182	0.2549	0.2434
Padded trigrams	0.2942	0.2197	0.2067	0.2144	0.2266
Bigrams	0.2460	0.2014	0.1838	0.2014	0.2021
Trigrams	0.2368	0.1853	0.1654	0.1830	0.1876
Edit distances	0.2064	0.1509	0.1242	0.1639	0.1555
Tanween-aware padded bigrams	0.1316	0.0997	0.0875	0.1066	0.0974
Tanween-aware padded trigrams	0.0452	0.0294	0.0294	0.0294	0.0294
Tanween-aware trigrams	0.0308	0.0225	0.0202	0.0202	0.0202
Tanween-aware bigrams	0.0722	0.0546	0.0500	0.0546	0.0546

performance. Clearly, therefore, although *tanween*-aware *n*-grams provide excellent effectiveness in which regular use of diacritics is ensured, they are not resilient to irregular diacritic usage.

Welch *t* tests show that the performance difference between ASOUNDEX and the best *n*-gram method, padded bigrams, is significant at an interval of 95% with a *p*-value of 0.0263; the difference between ASOUNDEX and edit distances is extremely pronounced, yielding a *p*-value of 1.73×105, and significant even at the 99% level.

In our final set of experiments, we fused string distance methods with ASOUNDEX. We utilized COMBSUM for this purpose, ranking retrieved results by the sum of scores returned by each individual technique. The fused techniques were tested over both queries with diacritics and those without. In Table 15, we illustrate the various combinations we considered, with results for testing over queries without diacritics. It shows that fusion of ASOUNDEX and edit distances results in an average precision of 0.6670 and *R*-precision of 0.4991, outperforming both plain ASOUNDEX and the fusion of ASOUNDEX with *tanween*-aware padded bigrams.

To confirm the significance of these results, we collected the results from all three of the scenarios described and tested their composite significance. The results of these tests are shown in Table 16.

TABLE 15. Effectiveness for fused techniques.

Technique	Average precision	*R*-precision
ASOUNDEX and edit distances	**0.6670**	**0.4991**
ASOUNDEX and *tanween*-aware padded bigrams	0.6556	0.4846
ASOUNDEX	0.5459	0.3787

From these results, we may draw the fact that although ASOUNDEX may be outperformed by edit distances and specialized *n*-gram methods in the specific instance in which regular use of diacritics is ensured, ASOUNDEX provides a statistically significant improvement over both techniques in the general situation. Also obvious is the fact that our fused techniques provide an improvement in performance that is significant even at the 99% level.

Analysis

Although ASOUNDEX does not necessarily outperform string-distance measures, it is far more resilient in terms of dealing with irregular use of diacritics. Also, in evaluating queries that made regular use of diacritics, *tanween*-aware *n*-grams far outperformed normal *n*-grams. Experiments with queries that do not make use of diacritics, however, showed the exact opposite result, as *tanween*-aware *n*-grams were stripped of their advantage. In both cases, an examination of the results yielded shows that *n*-gram techniques almost uniformly miss correct relevant names that are spelled differently but are still phonetically similar. This process is illustrated by the results retrieved for the query تهصين, a misspelled name that has three relevant results in the collection. Table 17 shows the top three names retrieved by the best-performing techniques when the query contains diacritics, as well as when diacritics have been stripped. Relevant names are underlined.

We find that ASOUNDEX retrieves two correct answers for queries with and without diacritics. Edit distances perform very well with queries that correctly use diacritics but deliver poor results when the query is stripped of diacritics.

TABLE 16. Significance of leading techniques.

Technique	Against padded trigrams (*p* value)		Against edit distances (*p* value)	
ASOUNDEX	0.00038	Significant	0.07118	Not significant
Edit distances	0.07327	Not significant	n/a	n/a
ASOUNDEX fused with *n*-grams	8.777e-09	Significant	4.544e-05	Significant
ASOUNDEX fused with edit distances	3.85e-09	Significant	2.586e-05	Significant

DOI: 10.1002/asi

TABLE 17. Top three results retrieved for the query name 'تَهْصين'.

Technique	Results for 'تَهْصين' (with diacritics)	Results for 'تهصين' (without diacritics)
ASOUNDEX	تَهَاني, تَحْسِينه, تَحْسين	تَهَاني, تَحْسِينه, تَحْسين
Edit distances	تَحْسِينه, تَهْنيد, تَحْسين	ذهين, حَصين, ثمين
Tanween-aware padded bigrams	عَهْدي, تَهْنيد, تَحْسين	حِصْن, أرين, أمين
Padded trigrams	تَحْسين, مَهدَين, تَهْنيد	تَحْسين, مَهدَين, تَهْنيد

The *tanween*-aware padded bigrams, the best *n*-gram technique when used on queries with diacritics, as do edit distances, fail to retrieve any relevant names when the query is stripped of diacritics. Padded trigrams, on the other hand, display their resilience to varied diacritic usage by returning the same results for queries with and without diacritics. The inconsistent performance of *tanween*-aware padded bigrams is caused by the value placed on diacritics by this technique. Because padded trigrams do not consider diacritics in any special manner at all, this technique is relatively unaffected by improper use. In addition to highlighting ASOUNDEX's resilience to varied diacritics usage, Table 17 reveals the importance of phonetic matching in an effective name search system: ASOUNDEX is the only technique that correctly returns the relevant names as the top results for this query. Edit distances incorrectly interpret 'تَهْنيد', as being relevant because, with transpositions being counted equivalently to insertions or deletions, the distance between this and the query is small. With the *n*-gram techniques, although padding improves performance, this particular query shows how relevant names that differ from the query name in the initial or terminal character can incorrectly be judged as relevant. The results also show that ASOUNDEX correctly retrieves two relevant names, but it misses a third, 'تَحَاسين', because our final ASOUNDEX algorithm categorizes the vowel *alif* as it would a consonant, causing it to be encoded differently than the query name is. Thus, although we have found that categorizing *alif*s as we do consonants improves performance in the general case, this particular query highlights a situation in which it would be advantageous not to perform this categorization.

The fact that edit distances outperform ASOUNDEX when thorough, proper use of diacritics is present in both queries and the collection is not unexpected, as Zobel and Dart (1996) state that edit distances are more accurate than phonetic matching techniques as approximate search tools. Zobel and Dart further suggest that our approach to formulating relevance judgments, which is based on a textual inspection of results rather than a phonetic evaluation of similarity, biases the judgments in favor of string-distance measures. We applied, on average, 2.29 character errors (substitutions, transpositions, etc.) to create each misspelled query. Our experiments confirm that edit distances yield high performance in terms of both recall and precision. However, their performance is reduced when thorough use of diacritics is not ensured. Table 18 shows results returned by

TABLE 18. Results for the query name 'سامْان' by ASOUNDEX and edit distances.

Technique	Top results returned
ASOUNDEX	سُعَيْدَان, سَامِعَة, سَامِع
Edit distances	سمْعَن, حَمْدان, سَمْعَان

ASOUNDEX and edit distances for the query name 'سامْان' when diacritics are used. Relevant results are underlined.

The use of edit distances retrieves both of this query's relevant names whereas ASOUNDEX retrieves none. This is because both of these relevant names are translated to the original query name with only two insertions or deletions. The ASOUNDEX code for each of the names retrieved differs from the query name's code by only a single character. Thus, they are preferred to the relevant names, which both differ from the query name more significantly in terms of spelling. In addition to their lack of resilience to improper diacritic use, note that the efficiency of edit distances is also an issue, as they are computed in quadratic time and, unlike ASOUNDEX and *n*-grams, must be computed at run time.

On the observation that string-distance measures provide high effectiveness and ASOUNDEX provides excellent resilience, we investigated pairing the techniques to capture the advantages inherent in each. Our results show that combining ASOUNDEX edit distances and with *tanween*-aware bigrams, the best of our *n*-gram techniques, results in the highest precision. Testing this combination over queries with and without diacritics showed that performance remains unaffected by mistakes in diacritic use. Similarly, combining ASOUNDEX with edit distances also yielded improved results. Using the same query previously shown to compare the top-performing techniques, Table 19 shows results obtained from our fusion experiments. Consistently with our previous results, two of three relevant results are retrieved, although the other result that neither ASOUNDEX nor our string distance measures were able to retrieve is still absent. The fused techniques are relatively resilient to varied diacritic usage.

We next consider our other example: the query 'سامْان', which we previously showed edit distances handled more effectively than ASOUNDEX. Table 20 shows the top results retrieved by our fused techniques.

Interestingly, the fusion of ASOUNDEX and *tanween*-aware padded bigrams does not retrieve any relevant results.

TABLE 19. Top three results retrieved by fusion techniques for the query name 'تَهْصين'.

Technique	Results for 'تَهْصين' (with diacritics)	Results for 'تهصين' (without diacritics)
ASOUNDEX and *tanween*-aware padded bigrams	تَسيم, تَحْسِينه, تَحْسين	تَهَاني, تَحْسِينه, تَحْسين
ASOUNDEX and edit distances	تَسيم, تَحْسِينه, تَحْسين	تَهَاني, تَحْسِينه, تَحْسين

DOI: 10.1002/asi

TABLE 20. Results for the query name "سامّان" by fusion techniques.

Technique	Top results returned
ASOUNDEX and *tanween*-aware padded bigrams	نعمان, سَامِعَة, سَامع
ASOUNDEX and edit distances	سمعَن, سَامع, سَمْعَان

Examining the results returned by each individual technique, we find that neither ASOUNDEX nor the *n*-gram technique retrieved any relevant results individually, a finding that explains this result. The fusion of ASOUNDEX and edit distances, on the other hand, retrieves both relevant names previously returned by edit distances.

Conclusions and Future Work

We developed two new algorithms for Arabic name search and compared them to existing language-independent techniques. ASOUNDEX is a phonetic technique based on English Soundex and deriving features from Editex and PHONIX. We have shown the value of phonetic matching in providing resilience over pure string similarity systems. Further, we have leveraged fusion of results to incorporate the specific strengths of both phonetic and string similarity techniques into our retrieval system, yielding effectiveness almost double that of language-independent *n*-gram techniques. For evaluation purposes, we have constructed a collection of 7,939 Arabic names along with sets of queries and relevance judgments.

Our results show that *R*-precision is superior to average precision for the Arabic name search task because the number of relevant results is too small for interpolation. However, weak ordering in the result sets demands that *R*-precision be calculated over a number of random permutations of the results. We also find reason to reexamine Zobel and Dart's (1995) observation that string similarity measures outperform phonetic matching techniques, noting that this is not true for Arabic.

In the future, we will explore several enhancements to ASOUNDEX, such as code shifting, described by Holmes and McCabe (2002), and other fusion methods. Further work is necessary to introduce greater resilience to transposition errors, which are more difficult to handle than errors such as substitution. Similarly, further improved resilience to improper diacritic use would likely continue to yield improvements for Arabic name search systems.

References

AbiFares, H. (1998). Comparison of Latin and Arabic scripts. Retrieved from http://www.sakkal.com/articles/Arabic_Type_Article/Arabic_Type3.html

Aljlayl, M., & Frieder, O. (2002). On Arabic search: Improving retrieval effectiveness via a light stemming approach. In Proceedings of the ACM Eleventh Conference on Information and Knowledge Management (pp. 340–348). New York: ACM.

Al-Shalabi, R., & Evens, M. (1998). A computational morphology system for Arabic. Workshop on Computational Approaches to Semitic Languages, COLING–ACL, 1998.

Angell, R., Freund, G., & Willet, P. (1983). Automatic spelling correction using a trigram similarity measure. Information Processing and Management, 19(4), 255–261.

Binstock, A., & Rex, J. (1995). Practical algorithms for programmers. Reading, MA: Addison-Wesley.

Borlund, P. (2003). The concept of relevance in IR. Journal of the American Society for Information Science and Technology, 54(10), 913–925.

Celko, J. (1995). Joe Celko's SQL for smarties: Advanced SQL programming (2nd ed.). Burlington, MA: Morgan Kaufmann.

Damerau, F. (1964). A technique for computer detection and correction of spelling errors. Communications of the ACM, 7(3), 171–176.

Dice, L.P. (1945). Measures of the amount of ecologic associations between species. Ecology, 26(3), 297–302.

Erikson, K. (1997). Approximate Swedish name matching survey and test of different algorithms (NADA Report TRITA-NA-E9721). Stockholm: Royal Institute of Technology.

Gadd, T.N. (1988). "Fisching fore werds": Phonetic retrieval of written text in information systems. Program—Electronic Library and Information Systems, 22(3), 222–237.

Gadd, T.N. (1990). PHONIX: The algorithm. Program, 24(4), 381–402.

Hall, P., & Dowling, G. (1980). Approximate string matching. ACM Computing Surveys, 12(4), 381–402.

Hodge, V., & Austin, J. (2001). Technical Report YCS 338. Department of Computer Science. York, England: University of York.

Holmes, D., & McCabe, M. (2002) Improving precision and recall for SOUNDEX retrieval. In Proceedings of the 2002 IEEE International Conference on Information Technology—Coding and Computing (pp. 22–28). Los Alamitos, CA: IEEE Computer Society.

Kang, B., & Choi, K. (2001). Two approaches for the resolution of word mismatch problem caused by English words and foreign words in Korean information retrieval. International Journal of Computer Processing of Oriental Languages, 14(2), 109–133.

Katzner, K. (2002). The Arabic language. Retrieved from http://www.worldlanguage.com/Languages/arabic.htm

Larkey, L., Ballesteros, L., & Connell, M. (2002). Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. In Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval (pp. 275–282). New York: ACM Press.

Levenshtein, V. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. Problems of Information Transmission, 1, 8–17.

McCabe, M.C., Chowdhury, A., Grossman, D., & Frieder, O. (1999). A unified environment for fusion of information retrieval approaches. In Proceedings of the 1999 Conference on Information and Knowledge Management (CIKM99) (pp. 330–334). New York: ACM Press.

Mizzaro, S. (1998). How many relevances in information retrieval? Interacting With Computers, 10(3), 305–322.

NCR. (1998). State of Texas selects NCR data warehouse solution to help it collect $43 million in additional taxes. News Release, May 18.

Pfeifer, U., Poersch, T., & Fuhr, N. (1995). Searching proper names in databases. In Proceedings of the Conference on Hypertext—Information Retrieval—Multimedia (pp. 259–275). Konstanz, Germany: Universitatsverlag Konstanz.

Pfeifer, U., Poersch, T., & Fuhr, N. (1996). Retrieval effectiveness of name search methods. Information Processing and Management, 32(6), 667–679.

Raghavan, V., Bollman, P., & Jung, G. (1989). A critical investigation of recall and precision as measures of retrieval system performance. ACM Transactions on Information Systems, 7(3), 205–229.

Rogers, H.J., & Willett, P. (1991). Searching for historical word forms in text databases using spelling-correction methods: Reverse error and phonetic coding methods. Journal of Documentation, 47(4), 333–353.

Shaw, J.A., & Fox, E.A. (1994). Combination of multiple searches. In Proceedings of the Text Retrieval Conference 2 (pp. 243–252). Greenburg, MD: National Institute for Standards and Technology.

Suwanvisat, P., & Prasitjutrakul, S. (1998). Thai-English cross-language transliterated word retrieval using Soundex technique. Proceedings

DOI: 10.1002/asi

of the National Computer Science and Engineering Conference, Thailand (pp. 210–213). Bangkok, Thailand: Thonbury University.
Tayli, M., & Al-Salamah, A. (1990). Building bilingual microcomputer systems. Communications of the ACM, 33(5), 495–505.
UBAccess. (2002). Hebrew and Arabic accessibility. Retrieved from http://www.ubaccess.com/hebrew-access.html
Wegener, S. (2000). Arabic and the WWW. Retrieved from http://ssgdoc.bibliothek.uni-halle.de/vlib/html/docs/arabwww_en.html
Zamora, E., Pollock, J., & Zamora, A. (1981). The use of trigram analysis for spelling error detection. Information Processing and Management, 17(6), 305–316.
Zobel, J., & Dart, P. (1995). Finding approximate matches in large lexicons. Software—Practice and Experience, 25(3), 331–345.
Zobel, J., & Dart, P. (1996). Phonetic string matching: Lessons from information retrieval. In Proceedings of the 19th International Conference on Research and Development in Information Retrieval (pp. 166–172). New York: ACM Press.

DOI: 10.1002/asi