

## TRANSLITERATED ARABIC NAME SEARCH

David Holmes                      Samsun Kashfi  
NCR Corporation  
Germantown, MD 20876  
e-mail: { david.holmes, samsun.kashfi}@ncr.com

Syed Uzair Aqeel  
Illinois Institute of Technology  
Chicago, IL 60616  
e-mail: aqeel@ir.iit.edu

### ABSTRACT

We address name search for transliterated Arabic given names. In previous work, we addressed similar problems with English and Arabic surnames. In each previous case, we used a variant of Soundex and n-grams to improve precision and recall of name matching compared against well known approaches such as the Russell Soundex algorithm. Unlike prior work, the proposed approach does not rely upon Soundex algorithms.

We experiment with combinations of n-grams of varying lengths. Our previous work focused on two character n-grams. As with our prior work, this approach uses standard SQL and remains portable to different relational database engines, demonstrated by implementing test in SQLAnywhere and Teradata environments.

### KEY WORDS

Name Search, SQL, Arabic Transliteration, n-grams.

### 1. Introduction

This paper presents a means of performing name search on transliterated Arabic given names and accounting for numerous orthographical variations. Addressing transliterated Arabic name search is a logical extension of our prior work on English and Arabic name search.

In addition to information retrieval, name search is important for some decision support systems that need to perform inexact matching. For example, multiple systems in the field of tax compliance perform name search to identify individuals and businesses that fail to file their taxes [0]. State tax agencies in Texas and Iowa have recovered hundreds of millions of dollars by identifying non-filing taxpayers. Data warehousing integration tasks, such as householding, also benefit from robust name matching capability. There are non-revenue related applications of the capability as well, such as criminal investigation and national security.

The system described herein uses SQL to apply a set of transformation rules, producing enhanced names used for searching. The rules reduce trivial orthographical variations. After rule application, the system parses

names into n-grams. The match process identifies related names by way of a similarity metric that counts shared n-grams. Rule derivation relied on domain knowledge of transliterated names and visual inspection of orthographical variations in the more common names.

Our test collection combines several lists of names taken from the internet. The goal was to build a good representative sample of given names and the collection used in this work represents Arabic, Turkish and Persian names. We believe the collection contains most variants and includes approximately 10-15% archaic names seldom used today.

We compared the test collection to a real world data set to validate comprehensiveness. In 2003, over 3 million people were registered with the United States Social Security Administration. Their names include orthographical variations, including 160 transliterated Arabic given names. Table 1 compares the top 2,000 given names with the subset of transliterated Arabic names.

Table 1. 2003 Name Distribution, US SSA

	Average Frequency All Names	Average Frequency Transliterated
Top Quartile	4,847	4,360
Second Quartile	816	796
Third Quartile	395	382
Bottom Quartile	228	225

In the United States, the distribution of transliterated names is very similar to the distribution of all names. While the collection may not have every single variant, it is reasonable to assume that it represents the great majority of real world transliterations. Considering the top 160 names account for all Arabic names registered in the United States with frequencies above 200 for one year, the remaining 5,600 names in the collection should cover most names used in the population.

This work builds upon prior work in a number of ways. The work removes the need for a Soundex algorithm in name search as it depends entirely upon n-grams. Secondly, the work uses a complete set of n-grams, where the best results from our prior works limited n-gram usage to bi-grams.

## 2. Prior Work

Fuzzy soundex was proposed for name matching by [1]. This work proposed the creation of multiple Soundex codes and bi-grams for each name in the collection. N-gram substitution rules removed many of the orthographical variations between names. The work proposed generating up to ten Soundex codes per name, using Russell, Celko and a new algorithm to create Soundex codes of lengths 3, 4 and 5 (and 2 for the new algorithm). An eleventh Soundex code supported a new technique, called code shift, dropping the second character position and shifted the rest of the code to the left. The code shift reduced insertion errors and increased recall.

Given a test name, the algorithm searched the collection for other names with common bi-grams and Soundex codes and used a Dice Coefficient [2] for a similarity metric. In their testing, Holmes and McCabe achieved best results by using Soundex codes and bi-grams.

The results improved upon Russell Soundex for three major reasons. First, by using three different Soundex algorithms, Russell, Celko [3] and Fuzzy, the approach increased recall to over 95%. Second, the approach could rank results by counting common features. Finally, n-gram substitutions, combined with n-gram scoring and multiple soundex algorithms produced the best results. Russell Soundex can identify candidate matches but offers no means of ranking results. The best experiment produced an average precision of 0.7071.

Parsing textual data and replacing strings using SQL was proposed by [4]. The work presents a SQL implementation of the well known Porter stemming algorithm [5]. The stemming applied n-gram substitution rules to words trimming their suffixes.

The database centric approach to parsing and information retrieval proved 94% linearly scalable using Teradata. The parsing used unchanged SQL and did not require any proprietary extensions to the language, offering a portable solution. Some of the examples in this paper include Sybase proprietary functions because the version SQLAnywhere used did not include a few SQL standard functions. The database centric approach is important for many of applications of this work, such as compliance because it integrates name search with structured data.

Techniques similar to [1] were written for Arabic by [6]. Soundex code generation used rules specifically

developed to handle nuances of the Arabic language, such as diacritical marks. Also similar to [1], the best average precision resulted from multiple ASoundex codes and tanween-aware padded bi-grams. The best experiment produced an average precision of 0.6662.

A somewhat different approach is described in [7], where English names were decomposed into phonemes, converted to pin-yin and transliterated into Chinese. The result is a set of name pairs equating an English name with a transliterated Chinese name. Transliterated syllables had an accuracy of 47.5%.

The work described in [8] is similar to [7] and uses trigrams during a language model training phase with a focus on reducing pin-yin error rates. The system uses the same data set, which has 3,875 English names. In the end, the system produces name pairs, equating English and Chinese names. The system reduced pin-yin error rates from 51.1% to 42.5%.

In [9], we find the motivation for substitution rules. Damerau identified the most common types of errors found in textual data: insertion, omission, transposition and substitution errors. "Errors" might be actual errors in the collection or legitimate but trivial variants of names.

## 3. Algorithms

There are three algorithms related to this work. The first governs how n-gram substitutions occur. The second defines how n-grams are parsed from names in the collection. The third algorithm defines name searches and how results are ranked.

### 3.1 N-gram Substitution Algorithm

The algorithm reads each name in the collection and applies each substitution rule where appropriate, as defined in Figure 1.

**Figure 1. N-gram Substitution Algorithm**

```

NGRAM-SUBSTITUTION ( collection, rules )
  for each name in collection
    do for each rule in rules
      do for i ← 1 to length [name]
        do if SUBSTRING ( name, i, j ) = rule
          then REPLACE ( name, i, j, rule )

```

The algorithm produces enhanced names that reduce trivial variations, examples of which are shown in Table 2.

**Table 2. Name Enhancement Examples**

Enhanced	Given Names
Kasim	Abul-Qasim, al-Kasim, al-Qasim, Kaseem, Kasim, Kassim, Qaseem, Qasim
Usama	Osama, Ossama, Ousama, Usama, Usamah
Ali	'Ali, 'Aliyy, Abu-'Ali, Aliyy, Alli, Allie, Aly,
Faruk	Faruk, Faruq, Farooq, Farouk, Farook

Table 3 defines the rules used in this work. Rule Rank is the precedence for applying rules with the most significant digit delineating groups of rules executed sequentially until no more modifications to the enhanced name occur. A flag indicates whether rules apply anywhere, to a prefix only or a suffix only.

**Table 3. N-gram Substitution Rules**

Rule Rank	Original N-gram ("_" = space)	Substitute N-gram	Suffix (S), Prefix (P), Anywhere (A)
1.001-1.006	al-, al_, el-, el_, abul, abu		P
2.001-2.003	-, ' , _		A
3.001-3.003	abdal, abdel, abdol	abdul	A
3.004	der	dur	A
3.005	q	k	A
3.006	allah	ullah	A
3.007-3.008	ean, ead	id	S
3.009	ai	ay	A
3.010	e	i	A
3.011	ou	u	A
3.012	aee	ay	A
3.013	o	u	P
3.014	ah	a	A
3.015	ae	ay	A
3.016	ei	ay	P
3.017	gh	k	P
3.018	kh	k	A
3.019	kah	ka	A
3.020	ie	i	A
3.021	awo	ao	A
3.022	awu	au	A
3.023	awz	az	A
3.024	dh	d	A
3.025	ou	k	A
3.026	kua	ka	A
3.027	aw	au	A
3.028	v	w	A
3.029	say	sy	P
3.30	g	j	P
3.31	sw	s	P
4.005	ee	i	A
4.015	oo	u	A
4.*	Doubles other than aa, ee, oo	reduce to single	A
5.001	ed	ad	S
5.002	el	al	S
5.003	eh	a	S
5.004-5.005	y, ii	i	S
5.006	iya	ia	A
5.007	ah	a	A
5.008	ry	ri	A
5.009	mo	mu	P
5.010	eya	ia	S

### 3.2 N-gram Generation Algorithm

The N-gram Generation algorithm, Figure 2, defines how n-grams are derived from names.

**Figure 2. N-gram Generation Algorithm**

```

NGRAM-GENERATION ( collection, rules )
for each name in collection
do for i ← 1 to length [name]
do for j ← 1 to length [name]
do sub ← SUBSTRING ( name, i, j )
INSERT-INTO-TABLE ( names_table, name)
INSERT-INTO-TABLE ( n_grams_table, sub)

```

An example of the algorithm output is given in Table 4. The example shows the complete set of n-grams for the unaltered and enhanced version of the transliterated name *Qadir*.

**Table 4. Complete N-gram Set Example**

Name	N-grams (Both, Original Only, Enhanced Only)
Qadir	a,ad,adi,adir,d,di,dir,i,ir,r, <b>Q,Qa,Qad,Qadi, Qadir,</b> <i>K,Ka,Kad,Kadi,Kadir</i>

### 3.3 Match and Rank Algorithm

Figure 3 defines the Match and Rank algorithm used by the name search process. This algorithm identifies candidate matches for a query name and increments the similarity metric whenever a feature is shared by the search name and the candidate match.

**Figure 3. Match Algorithm**

```

MATCH-AND-RANK ( query_name, collection )
for each name in collection
do score [name] ← 0
for each test-ngram in n-gram[query_name]
do for each collection-ngram in n-gram[name]
do if test-ngram = collection-ngram
then score [name] ← score [name] + 1

```

Given a test name, the match and rank algorithm compares the n-grams of the name to the n-grams for all other names. A Dice Coefficient using the common n-grams and the total n-grams for each name provides a similarity metric used to assess the quality and ranking of each match.

## 4. Implementation Details

This section describes the implementation of the aforementioned algorithms in a relational database. Given a set of names loaded to the database, an SQL extract, transformation and load (ETL) process applies the N-gram Substitution algorithm to the collection creating an enhanced name used in name search.

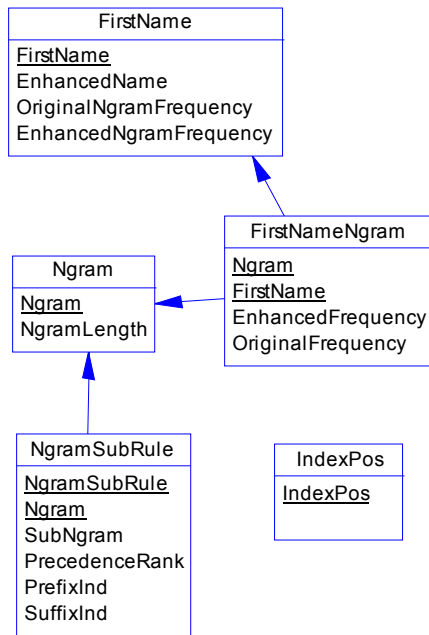
After the substitution process, another SQL step parses the original and enhanced names into their constituent n-grams, employing the N-gram Generation algorithm.

Finally, the Match and Rank algorithm is used to compare each test name to the collection and produce a ranked list of candidate matches. The output is compared to known relevant names and average precision is given for the test.

#### 4.1 Logical Data Model

The proposed system uses the data model described in Figure 4.

Figure 4. Name Search Data Model



The FirstName entity represents the domain of first names used in the system during the matching process. The FirstName attribute is the original, unmodified name. EnhancedName is the result of the substitution rules. The numeric attributes OriginalNgramFrequency and EnhancedNgramFrequency are counts of n-grams for the original and enhanced names.

The FirstNameNgram entity represents the domain of ngrams for each first name used in the system during the matching process. OriginalFrequency and EnhancedFrequency count the number of times given n-grams appear within names.

The NgramSubRule entity represents the domain of n-gram substitution rules used in the system during the name enhancement ETL process. The Ngram attribute is the pattern being searched for. SubNgram is the replacement pattern. PrefixInd and SuffixInd restrict the application of some rules to the beginning and ending of names.

The Ngram entity represents the total domain of ngrams. For name search, this is a logical entity and is not physically implemented.

The IndexPos entity represents a domain of integers ranging from 1 to 30, used in the n-gram substitution ETL process for parsing names into n-grams.

#### 4.2 N-gram Substitution SQL

Figure 5 presents the SELECT portion of the ETL process that applies the N-gram Substitution rules. Rules apply to prefixes, suffixes, or non-position sensitive n-grams. Table 3 shows the substitutions, which are applied in order to the enhanced names. Each rule rank, 1 through 5 is executed multiple times until no new substitutions occur.

There are a few key points to understanding this algorithm. The cross joins with IndexPos provide starting and ending positions to parse every n-gram from a name giving the ability to match substitution rules and adjust resulting enhanced names.

The PrefixInd and SuffixInd filters restrict rules to positions within names. The correlated subquery forces the rule application to the first occurrence within an enhanced name. Repeated execution of SQL results in rule application to every occurrence.

Figure 5. N-gram Substitution SQL

```

SELECT a.FirstName AS FirstName
      ,SUBSTRING(a.EnhancedName,1,b.IndexPos - 1)
      ||d.SubNgram
      ||SUBSTRING(a.EnhancedName,(1 + (b.IndexPos +
      c.IndexPos - 1)), LENGTH(a.EnhancedName) -
      b.IndexPos) AS EnhancedName1
FROM FirstName a
CROSS JOIN IndexPos b CROSS JOIN IndexPos c
INNER JOIN NgramSubRule d
ON SUBSTRING(a.EnhancedName, b.IndexPos,
c.IndexPos) = d.Ngram
WHERE ((d.SuffixInd = 0 AND d.PrefixInd = 0)
OR (d.PrefixInd = 1 AND b.IndexPos = 1)
OR (d.SuffixInd = 1
AND LENGTH(a.EnhancedName)
= (b.IndexPos + c.IndexPos - 1)))
AND a.EnhancedName LIKE '%'||d.ngram||%'
AND (b.IndexPos + c.IndexPos) < 30
AND LENGTH(a.EnhancedName)>LENGTH(d.Ngram)+1
AND b.IndexPos IN
(SELECT MIN(b1.IndexPos)
FROM FirstName a1
CROSS JOIN IndexPos b1 CROSS JOIN IndexPos c1
INNER JOIN NgramSubRule d1
ON SUBSTRING(a1.EnhancedName, b1.IndexPos,
c1.IndexPos) = d1.Ngram
WHERE (b1.IndexPos + c1.IndexPos) < 30
AND a.FirstName = a1.FirstName
AND d1.NgramSubRule = d.NgramSubRule
AND d1.NgramSubRule = @CurrRule )
GROUP BY FirstName, EnhancedName1
  
```

### 4.3 N-gram Generation SQL

The system parses a complete set of n-grams from the original and enhanced names. The process applies the count of original n-grams to the FirstName column OriginalNgramFrequency and the count of enhanced name n-grams to the EnhancedNgramFrequency column.

While the n-gram ETL process involves several SQL statements, the only complex operation is the n-gram parsing shown in Figure 6.

IndexPos is the domain of integers between 1 and the maximum character length of any name in the collection. The cross joins of two instantiations of IndexPos produce all possible starting and ending positions for n-grams with a name.

**Figure 6. N-gram Generation SQL**

```
SELECT SUBSTRING(a.FirstName
, b.IndexPos, c.IndexPos)
AS Ngram, a.FirstName, 1 AS NgramSource
, COUNT(*) AS OriginalFrequency
FROM FirstName a
CROSS JOIN IndexPos b CROSS JOIN IndexPos c
WHERE SUBSTRING(Ngram,c.IndexPos,1) <> ''
AND b.IndexPos BETWEEN 1 AND 30
GROUP BY Ngram, a.FirstName, NgramSource
```

### 4.4 Name Search SQL

The matching process finds potential similar names for a given search name with results ranked using a dual Dice Coefficient. Results are ranked in descending Dice Coefficient order for enhanced n-grams using the SQL shown in Figure 7. A descending Dice Coefficient for the original n-grams provides a secondary sort order. The SQL used for the match is as follows (EF = Enhanced Frequency, OF = Original Frequency, ENF = Enhanced Ngram Frequency, ONF = Original Ngram Frequency):

**Figure 7. Name Search SQL**

```
SELECT c.FirstName AS SearchName
, d.FirstName AS MatchedName
, SUM(CASE WHEN a.EF= b.EF THEN (a.EF + b.EF)
WHEN a.EF < b.EF THEN (2 * a.EF)
ELSE (2 * b.EF) END) AS EFCo
, (c.ENF + d.ENF) AS ETNF
, ((1.000 * EFCo) / ETNF) AS ESS
, SUM(CASE WHEN a.OF = b.OF THEN (a.OF + b.OF)
WHEN a.OF < b.OF THEN (2 * a.OF)
ELSE (2 * b.OF) END) AS OFCo
, (c.ONF + d.ONF) AS OTNF
, ((1.000 * OFCo) / OTNF) AS OSS
FROM FirstNameNgram a
INNER JOIN FirstNameNgram b ON a.Ngram = b.Ngram
INNER JOIN FirstName c ON a.FirstName = c.FirstName
INNER JOIN FirstName d ON b.FirstName = d.FirstName
WHERE a.FirstName = @MatchName
GROUP BY c.FirstName, d.FirstName, ETNF, OTNF
ORDER BY ESS DESC, OSS DESC
```

## 5. Results

Testing used a collection of nearly 5,819 Arabic first names transliterated into English. Testing searched the collection for relevant variants of 150 test names. Test names were selected without respect to the impact of substitution rules upon them. Substitution rules modified about 60% of the test names, similar to the whole collection that had 62% of its names modified by rules.

We evaluate results using standard measures of average precision and recall as described by [10]. Table 5 summarizes the collection demographics and test results.

**Table 5. Collection and Results Summary**

Description	Value
Collection Names	5,819
- Modified by substitution rules	3,590
- Distinct Enhanced Names	4,032
- N-grams by Name	173,097
- Distinct N-grams	42,607
Test Names	150
- Modified by substitution rules	90
Relevant Results	660
- <= 3 variants	68
- 4-5 variants	50
- >= 6 variants	32

Table 6 is a histogram of average precision for each test query for a couple of key tests. Testing bi-grams only is similar to our prior work, except it does not include Soundex. Also, we compare using all n-grams for original and enhanced names. Clearly, the substitution rules improve precision.

**Table 6. Average Precision Histogram**

Average Precision Range	Original N-grams Len=2	Original N-grams All	Enhanced N-grams All
1.00	7	10	85
0.9 – 0.9999	2	2	11
0.8 – 0.8999	8	7	32
0.7 – 0.7999	16	15	12
0.6 – 0.6999	19	23	4
0.5 – 0.5999	47	46	4
0.4 – 0.4999	24	22	2
0.3 – 0.3999	24	23	0
0.2 – 0.2999	3	2	0
0.1 – 0.1999	0	0	0
0.0 – 0.0999	0	0	0
Recall	0.9879	1.0000	1.0000
Avg Precision	0.5531	0.5622	0.9086

Table 7 demonstrates the importance of using rules to enhance precision and using combinations of n-grams to maximize recall.

**Table 7. N-gram Effectiveness**

N-gram Length	Average Precision	Total Recall	Pct Recall
Original Only			
1	0.4043	660	1.0000
2	0.5531	652	0.9879
3	0.5156	578	0.8758
All	0.5622	660	1.0000
Enhanced & Original			
1	0.7366	660	1.0000
2	0.8866	655	0.9924
3	0.8223	597	0.9045
1,2	0.8943	660	1.0000
1,3	0.9101	660	1.0000
1,4	0.9059	660	1.0000
1,2,3	0.9073	660	1.0000
1,2,4	0.9067	660	1.0000
1,3,4	0.9077	660	1.0000
All	0.9086	660	1.0000

Enhanced names also improve the performance of name searches that include a soundex match. Using the Soundex function built into SQLAnywhere, we add a filter requiring a search name and a result name to have a matching Soundex code.

**Table 8. Soundex Performance**

Test	Original	Enhanced
Average Precision	0.6904	0.9215
Total Recall	578	636
Pct Recall	0.8758	0.9636

Table 8 shows that matching on all n-grams and using an out-of-the-box Russell Soundex function in SQLAnywhere provides high precision, but does have some loss of recall and requires enhanced names for best performance.

The test environment was a Dell PC running Sybase SQLAnywhere version 9. There was no attempt to tune for performance. Even so, the one-time ETL process to build the database took about two hours and name searches averaged less than two seconds per search. Previous work [4] demonstrated near linear scalability of similar text parsing ETL using Teradata. In the previous work, an identical amount of text was parsed using small and large MPP server configurations and run time performance improve approximately linearly with increased hardware resources.

## 6. Conclusion

This work builds upon prior work in the area on name search. It uses some techniques that have previously demonstrated good precision and recall results. For example, including n-grams in the scoring algorithm

improve precision. Reducing trivial variations through n-gram substitutions also improved precision and recall. Both findings are consistent with [1] which studied name search against an English language collection.

In this work, as in [1], there are only a few substitution rules and the rules apply to many names. The purpose of the research was to develop a concise set of generic rules, not a set tailored to a particular set of test names. As shown above, the rules altered 60% of the test names and 62% of the names in the whole collection, reducing orthographical variations by about 30%. After rule application 4,032 distinct “enhanced” names remained from the set of 5,819 original names.

Also, [6] showed the importance of n-grams, in the Arabic language, in scoring results, but did not address n-gram substitutions beyond tanween-aware padded bi-grams.

In other words, the similar approaches have proven applicable in English, Arabic and Arabic transliterated into English. The rules are different, but n-grams are an integral part of name search in each case because they are the foundation for the similarity metric. N-grams allow name search to overcome typical errors and variants found in names and classified by Damerau, such as insertion errors, because n-grams allow for a partial match. In this work, n-grams of all lengths alone are used for name search, removing the dependency upon Soundex algorithms used in the previous work.

While utilizing Soundex functions improved precision slightly, we also observed some fallout in recall. This is consistent with our previous work on English and Arabic Soundex functions.

We also demonstrated that a complete set of n-grams slightly outperformed a bi-gram only approach with respect to precision. Bigrams alone outperformed trigrams, the second most precise n-gram. Uni-grams had no fallout, but were not good for ranking purposes when used in isolation. Uni-grams guarantee very high recall and longer length n-grams improve precision. Uni-grams, used with almost any combination of n-grams taken from enhanced names, produced similar precision and recall results.

Our approach is somewhat different from other approaches to handling transliterated names. For a given test name, our work produces a candidate list and assigns each name a similarity score. Higher scores are more likely to be relevant, but there is not an absolute assessment. In the work done by Meng [7], names are deemed relevant when they are in a name pair, otherwise, a search result is considered not relevant.

The proposed system uses SQL to perform all tasks. Implementations on Sybase SQLAnywhere and NCR

Teradata show the widest range of database portability from small devices to the largest massively parallel processing servers. Using multiple server configurations, [4] demonstrated the near linear scalability of text parsing using SQL and the Teradata database. Another advantage to the proposed system is performance. Prior work using Soundex and N-grams requires more processing than the proposed system that uses only n-grams. The match performance improves because the match SQL performs two fewer joins to a Term-Soundex table.

Applying this approach to other languages is an area for future work. Also, the current match process tends to have some ties in the similarity measure. In SQL, the descending ORDER BY clause randomly sorts the tied scores, resulting in minor fluctuations, less than 0.01, in average precision.

### References:

- [0] NCR. State of Texas Selects NCR Data Warehouse Solution to Help it Collect \$43 Million in Additional Taxes, *News Release*, 1998.
- [1] D. Holmes, M. McCabe. Improving Precision and Recall for Soundex Retrieval. *IEEE International Conference on Information Technology: Coding and Computing (ITCC)*, 2002.
- [2] L. Dice. Measures of the Amount of Ecologic Associations Between Species. *Journal of Ecology*, 26, 1945.
- [3] J. Celko, *Joe Celko's SQL For Smarties: Advanced SQL Programming*. (Morgan Kaufmann Publishers, Inc., 1995).
- [4] D. Holmes. SQL Text parsing for Information Retrieval. *Conference for Information and Knowledge Management (CIKM)*, 2003.
- [5] M. Porter. An algorithm for suffix stripping. *Program*, 14(3), 1980.
- [6] S. Aqeel, E. Jensen, S. Beitzel, D. Grossman, O. Frieder. On the Development of Name Search Techniques for Arabic. *To appear, JASIST*, 2003.
- [7] H. Meng, et al. Generating Phonetic Cognates to Handle Named Entities in English-Chinese Cross-Language Spoken Document Retrieval. *Proceedings of ASRU*, 2001.
- [8] P. Virga, S. Khudanpur. Transliteration of Proper Names in Cross-Lingual Information Retrieval. *SIGIR*, 2003.
- [9] F. Damerau. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM* 7, 1964.
- [10] D. Harman. Proceedings of the Third Annual Text Retrieval Conference (TREC). *TREC-3*, 1994.