Rethinking Full-Text Search for Multilingual Databases

Jeffrey Sorensen and Salim Roukos
IBM T. J. Watson Research Center
Yorktown Heights, New York
<sorenj|roukos>@us.ibm.com

Abstract

Textual fields are commonly used in databases and applications to capture details that are difficult to formalize—comments, notes, and product descriptions. With the rise of the web, users expect that databases be capable of searching these fields quickly and accurately in their native language. Fortunately, most modern database systems provide some form of full-text indexing of free text fields. However, these capabilities have yet to be combined with the simultaneous demand that databases provide support for world languages. In this paper we introduce several of the challenges for handling multilingual data and introduce a solution based on an architecture that enables flexible processing of texts based upon the properties of each text's source language. Extending the indexing architecture, and standardizing the query capabilities, are important steps to creating the applications that will serve world markets.

1 Introduction

Text fields capture *unstructured information* such as descriptions, comments, notes and other difficult to formalize information, and are part of most large scale database applications. Standard database tools, and the popular Structured Query Language (SQL), provide little support for applications that demand indexing and searching these fields.

Even so, full-text search indexing is a *standard* non-standard component of every major database system, including the popular open-source databases MySQL and PostgreSQL, through add-ons such as TSearch2 [11, 3]. Microsoft SQL Server provides the ability to generate full-text indexes and, as shown in Figure 1, a query syntax for searching the database records.

These three examples illustrate searching for a keyword, searching for the different inflected forms of the word foot (feet, footed, etc.), and searching for documents that contain two terms in close proximity. More traditional databases do not provide such query capabilities. While most databases provide string wildcards and LIKE operator or regular expression matching, using them for search of text fields requires laborious record by record string comparison operations.

In order to implement fast full-text search, a *full-text index* must be built. However, to make a full-text index one must understand issues of tokenization, punctuation handling, normalization stemming and morphology, and word segmentation; these are all properties that are specific to each language and region. Today, most databases are configured to contain only one language, but for international companies this is already changing.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

```
SE AdventureWorks;
GO
SELECT Name, ListPrice
FROM Production.Product
WHERE ListPrice = 80.99
AND CONTAINS(Name, 'Mountain');
GO

USE AdventureWorks;
GO
SELECT Comments, ReviewerName
FROM Production.ProductReview
WHERE CONTAINS (Comments,
    'FORMSOF(INFLECTIONAL, "foot")');
GO

USE AdventureWorks;
GO
SELECT Description
FROM Production.ProductDescription
WHERE CONTAINS(Description,
    'bike NEAR performance');
GO
```

Figure 1: Examples of full-text queries for Microsoft SQL Server 2005 [5]

We distinguish between (1) a *single language database*, (2) a database that contains specific fields in a specific language (as in a system that supports searching Arabic and Chinese documents using English queries), and (3) the most general case, a database with fields that have multiple languages in the same field. We are proposing to use an architecture that supports all three types of databases.

Both Unicode support and full-text indexing are recent additions to traditional relational databases. However, as the Internet becomes truly international, see Figure 2, the ability to index world languages will become essential. Especially note that the size of the "Other" category means that developers will need to handle *many* new languages.

Looking at Microsoft SQL Server, again as representative of the state of the art, Microsoft uses a separate operating system component to implement the full-text search. This component has some support [5] for international text support using "word breakers, stemmers, and filters." However, this fixed architecture, originally designed for English, makes it difficult to handle databases that have fields from multiple languages.

In this paper we will be describing in detail the differences between the processing needed for several languages. By discussing languages that differ greatly in morphological complexity, we hope to demonstrate the need for a flexible, modular architecture for full-text indexing. To address this we propose database developers look to the Unstructured Information Management Architecture (UIMA) project affiliated with the Apache web server and its related projects.

2 Related Work

This paper references a number of papers dealing with various aspects of text processing needed for constructing indexes. However, the details of how to build and maintain indexes are beyond the scope, but well summarized in [29]. More recent developments in building compressed full-text indexes are presented in [16].

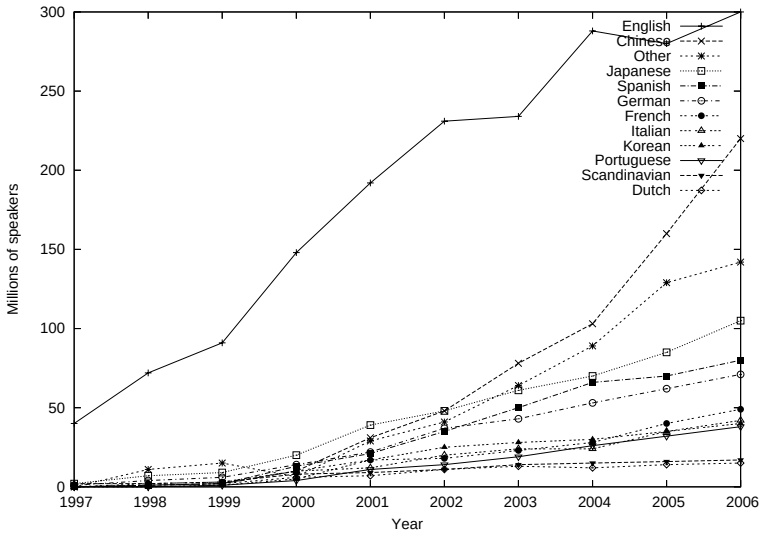

Figure 2: Internet population growth by language [24]

Our focus is primarily on the operations required to extract terms from the text for indexing purposes. However, there are several papers related to the building of indexes and their use in search. In [13], query operators are built into a database engine to allow fuzzy matching of strings from different languages through a mapping to approximate pronunciation differences, and, more abstractly, using a multilingual taxonomy, semantic category matching. Similarly, [32] studies the application of various edit distances seeking to match strings cross-lingually if their pronunciations are similar.

The trade off between building an exhaustive index and using a fuzzy distance metric during search involves trade-offs between computational complexity and memory storage. Several algorithms are compared in [21], matching terms cross lingually based upon string matching functions. Algorithms of this type can be used to reduce the total size of the index by collapsing interlanguage spelling variation. Similarly, [1] presents work that seeks to identify terms in documents that represent names of people or places in different languages.

3 Internationalization of Applications

Traditionally, internationalization efforts have focused on creating applications that can be operated by populations in different languages. This means tailoring fonts, menus, and other application messages often through the modification of global operating system settings. Many systems today determine the precise manner to collate and search strings through global “locale” settings. While these technologies have allowed products to be sold in international markets, this approach to “internationalizing” applications has been largely unsuccessful as an approach to creating multilingual applications. Even today, the standard ANSI C++ language has very limited support for international character representations.

Web browsers represent the leading edge of internationalization technology, as the web rendering technologies are the only software currently capable of rendering multilingual texts irrespective of most operating system settings. To a large extent most users today can effectively retrieve, and even create, texts in most languages, often limited only by their keyboards and the availability of alternate text input methods. This is amply demonstrated by the Wikipedia project [28] which, using a unified set of web tools, has successfully grown to more than 10,000 articles in each of 21 languages.

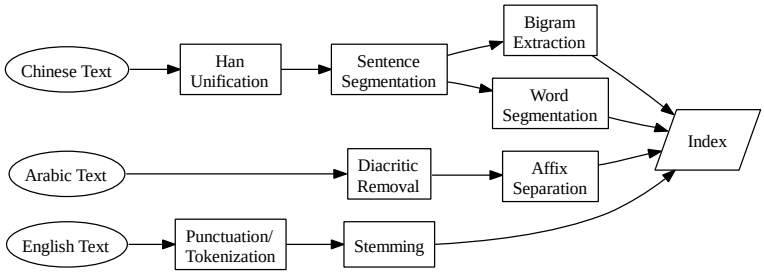

Figure 3: Typical text processing components for indexing

4 Modularization of Text Analysis

Developing international applications is a challenge for any platform. At present, implementing applications with full-text search requires correctly configuring a database server's operating system and a careful design of the database. If full-text searching is to become a standard feature of applications, making it work with multilingual texts must be simplified. Some of the current impediments include: a lack of standards for indexing and query specification, small alphabet assumptions in the text-processing components of databases and operating systems, the widespread use of components that do not correctly transfer Unicode data.

We seek to address many of these problems using a text processing architecture that is sensitive to the source language, even if said language is not specified explicitly. We present examples of processing pipelines for several languages that represent some of the widely varying needs.

4.1 Text-Indexing Pipeline

The full-text processing pipeline, as implemented in all of the current indexers for current database platforms, invariably involves a word segmenter, often just synonymous with "whitespace" tokenization. Many systems also make use of *stemmers* and *stop word lists*. Such a pipeline is usually adequate, at least for English texts. This is less true for other European languages which often have much richer word morphologies.

The Unstructured Information Management Architecture (UIMA) is a set of standards [9] and an Apache hosted collection of libraries intended to foster interoperability between developers of semantic analysis tools and other components for document search. While UIMA is intended to integrate analysis tools that reach to higher levels of abstraction than is typical for full-text search [17], it also has considerable strength for internationalization through its integration with Java and the International Classes for Unicode (ICU) [12] project. The capability to chain a sequence of text analysis engines, when appropriate, is a crucial capability when dealing with multiple languages. Figure 3 illustrates several chains of analysis tools used for a variety of languages.

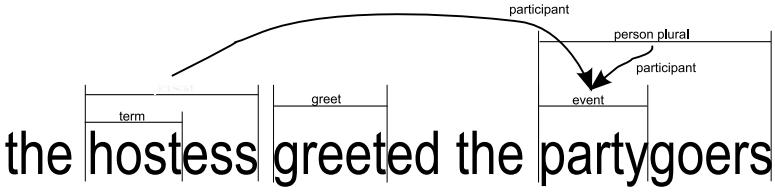

Figure 4: Marking character spans using stand-off notation in UIMA

4.2 Stand-off Notation

UIMA, through the Common Analysis Structure (CAS) represents tokens and other values to be indexed through the use of stand-off notation. Stand-off notation is the separate demarkation of text substrings via data structures that mark the begining and ending character offsets, as opposed to mark-up which uses *in-line* notation which is inserted into the text document. As the original text is unmodified, it can be inspected by any indexing processing subsystem. Stand-off notation is also important when text spans may be overlapping. The case of Chinese is exemplary, as Chinese text must be broken into individual words either using rules or some other automatic mechanism. However, this cannot be done unambiguously. Therefore, a common practice is to index Chinese simultaneously as words and as *overlapping bigrams* [14]. By referring to the text through a sequence of offset pairs instead of a single sequence of tokens, stand-off notation can support indexing of these overlapping spans. Figure 4 illustrates a potential use of stand-off notation to mark an English sentence that contains complex morphology. In addition to finding word stems and labeling verbs with their infinitive form, we may also apply, potentially, overlapping labels of semantic categories like "persons" or "locations" and even hierarchical relationships between these labels (e.g., the PARTICIPANT(PERSON,EVENT) relation in Figure 4).

4.3 Subjects of Analysis

When documents are translated, either by human or machine, or when text processing is used in a manner that modifies text in an irreversible way, the CAS is capable of maintaining multiple parallel representations. Within UIMA, these separate representations are considered differing *views* of the same text. Multiple subjects of analysis can be an important tool when building an index multilingually, as links into a view provide the matching correspondence in the original text. Thus both the document and its translation may be maintained in parallel in this CAS container.

4.4 Unicode Representation

The Unicode standard is an ambitious effort to represent the union of the characters from all written languages into one extended alphabet. Unicode remains controversial as a solution in some quarters, particularly among Chinese and Japanese communities [27]. The controversy is in part due to the complexity of the task of cataloging all of the hundreds of thousands of characters that occur in languages around the world. However, the problem is also due to the difficult experiences that developers have had with incomplete or, often, incorrect implementations of the Unicode Standard.

The Unicode consortium has been very active and the Unicode Standard is a rapidly evolving document with changes in virtually every aspect. The reference, and open-source, implementation [12] of the standard - the International Classes for Unicode (ICU) project has been under constant evolution and is itself a large and complex project that is often included, in whole or in part, in other projects such as Apache and the Java language.

5 Text Analysis Components

Most full-text indexing systems have a fixed architecture for text ingestion. At present, full-text indexing is still considered a special feature with only limited integration into the database. This means, in part, that the database has very limited control over the processing of individual text fields.

We are proposing that a more flexible approach be used. To illustrate why this would be beneficial, we would like to review some of the major components of text analysis to demonstrate how their relative importance changes, depending upon which language is being processed. When these multiple components can be combined flexibly on a record by record basis, perhaps based upon the content of other fields, multilingually indexing

and searching can be more easily and effectively accomplished. This dynamic ability to combine text-analysis components is one of the principal design goals of the UIMA project.

5.1 Language Identification

Where possible, applications should allow users to specify unambiguously what language a source text is written in. Typically, users enter texts in the same language as the interface an application is written in. However, this is a brittle assumption for polyglots. For text entry, the ability to check spelling is highly desirable, and this requires specification of not only the language but even the specific regional variant. Well written applications allow the user to make these specifications explicitly, although support for mixed languages within a single text field can pose significant interface challenges. When users cannot specify the language, or in cases where text fields are otherwise uncategorized, statistical techniques can be employed to determine the likely language [23, 2]. The UIMA standoff annotation can effectively encode the language ID of different spans in mixed language fields.

5.2 String Matching

The question of whether two strings, perhaps one typed in by the user and one stored in a database, are equivalent is an essential component of nearly every application. While less complicated than the issue of *near match* or *phonetic match*, there are difficulties that must be faced when a single string has multiple representations that are visually equivalent.

In the English world, many applications make use of lower case comparisons so that "English" and "english" are considered equivalent. However, this is quite problematical for applications comparing "espanol" with "Español" for a number of reasons, including ongoing difficulties associated with the case folding functions and case-insensitive string comparison functions that are based on ASCII character assumptions.

The Unicode standard has a substantial literature on the problem of accented characters, the first to be aware of is the existence of the variety of *normalization forms* [7] where characters can be represented in both composed and decomposed forms. In addition, the Unicode report on *Case Mappings* [8] gives some guidance on how to perform a case-independent string matching.

The growth of web search as a user interface has greatly changed the expectations of users. Most search engines use broad equivalence classes so that distinctions between searches that involve punctuation and accents are often impossible to make. This can be both forgiving when the representation is ambiguous (consider résumé versus resume) but maddening when the difference is essential.

However, when considering languages outside of European contexts, these problems are only the beginning. Although most non-European languages do not have upper and lower case distinctions, some languages have rich morphology that makes simple many-to-one equivalence insufficient for searching indexes. We will consider two specific languages cases as illustrative.

5.2.1 Traditional and Simplified Chinese

For databases and user interfaces, Chinese presents numerous challenges. Beyond the sheer complexity and size of the character set, there also exist several related language variants. In Taiwan and Hong Kong, *Traditional Chinese* characters are the predominant written form. Mainland China and Singapore use the much more recently developed *Simplified Chinese* character set.

The Unihan database is a project that seeks to index the relationship between the characters used in these different representations. While it important to maintain the distinctions between the various forms of the written Chinese language for rendering purposes, for search and indexing the representation of terms should be unified, typically by mapping to simplified Chinese. Of course, what constitutes a "term" in Chinese, where characters are written without any white-space separation, is a matter addressed in Section 5.3.3.

ٱلمقر	مقرهم
ٱل + مقر	مقر + هم
aal + maqar	maqaru + hum
the location	their location
بِٱلمقر	بمقرهم
ب + ٱل + مقر	ب + مقر + هم
bi + aal + maqar	bi + maqari + hum
by the location	by their location
لِلمقر	لمقرهم
ل + ٱل + مقر	ل + مقر + هُم
li + aal + maqar	ly + maqari + hum
to the location	to their location

Figure 5: An illustration of Arabic morphological changes

Many older string handling libraries assume byte or smaller alphabet sizes - in particular libraries that perform compression; many Asian texts clearly violate these assumptions. In addition to the some 70,000 unique ideographs designated in the Unicode specification, there exists extension mechanisms such as *ideographic descriptions* which allow users to describe an individual character using a hierarchical combination of other characters; a description that may be as many as 16 Unicode code points, meaning a single rendered character might be described by some 30 bytes in a file.

Character descriptions that are this long and complex are often chopped into meaningless sequences by naïve components. This is one of the many reasons we advocate, wherever possible, the use of *standoff notation* for text processing components.

5.2.2 Arabic

The Arabic alphabet is more similar in size to English than Chinese. However, it is much richer than either language in morphological complexity. In written Arabic, a single word often represents many terms, sometimes related. Written Arabic does not typically encode short vowel sounds; the reader determines the meaning based upon the context, filling in the missing vowels. Through the use of *diacritics*, or accent marks, the short vowels can be specified, and often are, in contexts where they must be specified unambiguously such as in children's educational books or poetry.

Unicode supports the encoding of diacritics through combining forms and through the use of composite forms for the most common diacritic combinations. That is, there is more than one representation of many accented characters, and they are graphically equivalent. This is true for other languages besides Arabic. For example, the "ü" character may be equivalently encoded using the code point `U+00FC`, or as the ASCII equivalent letter "u" `U+0075` followed by the combining diaeresis (or umlaut) character `U+0308`. Note too that this is also distinct from the stand alone diaeresis character `U+00A8`.

For Arabic, diacritics are not used for searching and indexing because they do not typically appear in the written text. However, a much larger problem exists with Arabic writing, and that involves the complex morphology. Arabic words typically combined with suffixes and prefixes that indicate gender, number, possessives and other relationships typically represented with prepositions in English. Searching typically requires identifying the "root" word or, for more effective searching, the "stem." Figure 5 shows an example of these types of modifications all of which share the same stem word.

5.3 Segmentation

Working with international or multilingual texts also presents problems already familiar to practitioners working with English texts. Consider the issue of *tokenization*, or the practice of segmenting an English text into word or word-like units.

One of the first difficulties that developers encounter is how to correctly handle punctuation. A first approximation is to simply consider words to be all contiguous alphabetic sequences. However, the frequent use of abbreviations, such as "Mr." or "Mrs." would be incorrectly segmented as a sentence boundary by a simple rules based tokenizer that considered periods as end of sentence markers.

In many cases, the solutions used for English tokenization can be applied to other languages directly. We will review some of those techniques in the subsequent sections. However, it is important to also note the ways in which techniques originally developed for other languages are now influencing the processing of English texts.

5.3.1 Rule based segmenters

The highly popular Snowball stemmers [22], originally developed for the English language, are used for a variety of European languages. Snowball is a language for encoding an algorithmic description of a stemming algorithm as well as C++ and Java code generators that convert the algorithmic description into code. The Snowball website includes ample references detailing the effect that stemming has on information retrieval.

Unfortunately, rule based segmenters, at least as they are used today, are limited in the amount of context that can be employed when trying to determine the correct word root. Words which involve spelling changes can be incorrectly processed; "mouse" being the root of "mice" is a well known example. As one begins to integrate texts with increased use of morphological changes, the limitations of rule based systems become increasingly apparent.

5.3.2 Statistical Stemmers

In addition to the rule based systems popular for European languages, the use of statistical models has received considerable attention [15] for languages such as Arabic. In these cases, character or word n-grams are used with a dynamic programming search algorithm to find the most likely segmentation of words into roots, prefixes, and suffixes (affixes).

Arabic segmentation can also depend upon contextual features not captured by n-grams, and in these cases additional features may prove useful [31]. However, it is important to remember that search terms are supplied without context, and determining which index entries match can be quite difficult when compared to the words in context. In these cases, query expansion or n-best segmentation should be applied to make the search more inclusive.

5.3.3 Words

Text processing of documents in multiple languages often entails solving problems that English speaking developers take for granted. For example, in many Asian languages, the concept of a *word* is subtle. Chinese and Thai are written in a style that does not delimit word boundaries.

The Unicode consortium has acknowledged this problem, and has sought to provide applications such as word processors with reasonable equivalents of the "next word" and "last word" commands through the use of dictionary based techniques. However, the current implementation, which is only currently used for Thai, is still considered by the community to be a work in progress.

The segmentation of Chinese into word units, such as those that denote actions, places, or people, is an area of considerable research [26] among the machine learning community. As with segmentation, the use of

statistical machine learning techniques show the greatest promise for providing an automatic means of separating words in Chinese and, potentially, other Asian languages.

5.3.4 Transliteration

Representing names of people and locations in non-native languages requires a system of mapping from one representation to another. Usually this is done phonetically, although in many cases place names are simply different words in different languages requiring, in this case, translation dictionaries. Figure 5 includes several such examples of Arabic words rendered to be read phonetically, using the letters of the English language.

Phonetic transliteration is also very commonly used in Chinese names, often resulting in letter sequences atypical of English rules, such as the name "Qian." Transliteration is a process that is amenable to automatic methods [25]. It should also be noted that ICU has support for rule based transliterations [12] even though this is not part of the Unicode standard.

5.4 Translation

Machine translation has been demonstrated [30, 10, 6] to be effective in providing cross-lingual search. Using the CAS Subject of Analysis capability, it is possible, in a UIMA processing pipeline, to simultaneously maintain multiple representations of the same text. By indexing records in both their source language and one or more translations that can be indexed in parallel, thus making it possible to build cross-lingual search tools.

5.5 Semantic Labeling

The information retrieval community also performs higher levels of annotation of texts, including the identification of "entities" such as people, places, organizations, and companies. The extraction, from text fields, of these types of objects—and the ability to build structured data from free text [4] also promise to greatly enhance future search capabilities. Thus, it is important for database systems to include flexibility in their text analysis architecture so as to not limit future applications' ability to use such deeper analysis.

6 Information Retrieval for International Text

Extracting terms from text fields is only the first step in implementing full-text search, as the terms must be placed into an index that provides a fast reverse map from terms back to the relevant documents. The details of building such an index can range from straightforward to complex, depending upon the quantity of text and the vocabulary. While many of the issues are beyond the scope of this paper, we would like to introduce several relevant ideas from the information retrieval community that can be employed to make index based retrieval more valuable.

6.1 String Comparison

Approximate string matching techniques, such as the use of edit distance or phonetic matching is an important tool for assisting users when the exact spelling of a term is unknown. Efficient computational methods exist [20] for many edit distances. However incorporating fuzzy string matching in the index search has to be carefully throttled to mitigate the increased computational requirements.

6.2 n-gram Indexing

Chinese text, in particular, due to its lack of explicit word segmentation is frequently indexed [14] using character bi-grams. That is, every overlapping two character sequence is placed into the index. Interestingly, as we move to

Figure 6: A demonstration of a cross-lingual search engine

larger n-grams, this technique can also be used for European languages. In [18], character n-grams were found to be comparable to other text indexing techniques. Although similar results were not found in the case of Arabic [19]. Ideally, n-gram enabled search can be combined, in a weighted fashion, with traditional word based search and provide near matches in cases where exact matches are not found, without introducing expensive string comparisons across the entire database contents.

7 Conclusion

Full text indexing typically involves the process of stripping away details that are unimportant in terms of search. However, complete applications must also report results and generate meaningful output. Figure 6 shows an example of output from a cross-lingual search engine that has found relevant Chinese and Arabic documents given search terms in English. The system uses UIMA and many of the components discussed in this paper to provide access to a multilingual database of documents.

Fully indexed text fields in databases often change substantially the way that applications are designed. While structured data, and database normalization are important contributors to efficiency and scaling, many real world problems resist structuring. Text fields are widely used in applications to capture those *ad-hoc* details. Without full-text indexing, these fields are essentially opaque to the application, and, unfortunately, the users.

As applications continue to push into international forums, the need to uniformly handle records regardless of the language is increasingly important. Fortunately, mature standards already exist that database designers should consider when extending these capabilities. Unicode technology is already part of most desktop computers through their web browsers. Modularizing the full-text indexing capability of contemporary databases, and standardization of the analysis would be a substantial step in the right direction.

References

[1] Yaser Al-Onaizan and Kevin Knight. Translating named entities using monolingual and bilingual resources. In *ACL '02: Proc. of the 40th Annual Meeting on Association for Computational Linguistics*, pages 400–408, Morristown, NJ.

[2] Olga Artemenko, Thomas Mandl, Margaryta Shramko, and Christa Womser-Hacker. Evaluation of a language identification system for mono- and multilingual text documents. In *SAC '06: Proc. of the 2006 ACM symposium on Applied computing*, pages 859–860, New York, NY.

[3] Oleg Bartunov and Teodor Sigaev. Tsearch2. PostgreSQL RDBMS Extension http://www.sai.msu.su/ megera/postgres/gist/tsearch/V2/.

[4] Jennifer Chu-Carroll, John Prager, Krzysztof Czuba, David Ferrucci, and Pablo Duboue. Semantic search via XML fragments: a high-precision approach to IR. In *SIGIR '06: Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 445–452, New York, NY.

[5] Microsoft Corporation. SQL server 2005: Full text search architecture. http://msdn2.microsoft.com/en-us/library/ms142541.aspx, April 2006.

[6] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust HLT applications. In *ACL '02: Proc. of the 40th Annual Meeting on Association for Computational Linguistics*, pages 168–175, Morristown, NJ.

[7] Mark Davis. Unicode normalization forms. Unicode Technical Report 15, The Unicode Consortium, San Jose, CA, August 1998.

[8] Mark Davis. Case mappings. Unicode Technical Report 21, The Unicode Consortium, San Jose, CA, November 1999.

[9] David Ferrucci and Adam Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, 2004.

[10] Martin Franz, J. Scott McCarley, and Salim Roukos. Ad hoc and multilingual information retrieval at IBM. In *TREC '89: Text Retrieval Conference*, National Institute of Standards and Technology, pages 104–115, 1998.

[11] J. M. Hellerstein, J. F. Naughton, and Avi Pfeffer. Generalized search trees for database systems. In *Proc. of the 21st International Conference on Very Large Data Bases*, 1995.

[12] International Business Machines, Inc. International components for Unicode user guide. Software Library http://icu.sourceforge.net/, 2006.

[13] A. Kumaran, Pavan K. Chowdary, and Jayant R. Haritsa. On pushing multilingual query operators into relational engines. In *ICDE '06: Proc. of the 22nd International Conference on Data Engineering*, 2006.

[14] K. L. Kwok. Comparing representations in Chinese information retrieval. In *Research and Development in Information Retrieval*, pages 34–41, 1997.

[15] Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. Language model based Arabic word segmentation. In *ACL '03: Proc. of the 41st Annual Meeting on Association for Computational Linguistics*, pages 399–406, Morristown, NJ.

[16] V. Mäkinen and G. Navarro. Compressed full-text indexes. Technical Report TR/DCC-2006-6, Department of Computer Science, University of Chile, April 2006.

[17] Alan Marwick. Text mining for associations using UIMA and DB2 Intelligent Miner. *IBM Developer Works*, http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0602marwick/, February 2006.

[18] Paul Mcnamee and James Mayfield. Character n-gram tokenization for European language text retrieval. *Inf. Retr.*, 7(1-2):73–97, 2004.

[19] Suleiman H. Mustafa and Qasem A. Al-Radaideh. Using n-grams for Arabic text searching. *J. Am. Soc. Inf. Sci. Technol.*, 55(11):1002–1007, 2004.

[20] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[21] Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Järvelin. Fuzzy translation of cross-lingual spelling variants. In *SIGIR '03: Proc. of the 26th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 345–352, New York, NY, 2003.

[22] M. F. Porter. Snowball: A language for stemming algorithms. http://snowball.tartarus.org, 2001.

[23] John M. Prager. Linguini: Language identification for multilingual documents. In *HICSS '99: Proc. of the Thirty-Second Annual Hawaii International Conference on System Sciences-Volume 2*, page 2035, Washington, DC, 1999.

[24] Global Reach. Global Internet statistics (by language). http://global-reach.biz/globstats/evol.html, 2004.

[25] Charles Schafer. Novel probabilistic finite-state transducers for cognate and transliteration modeling. In *Proc. of the 7th Conference of the Association for Machine Translation of the Americas*, pages 203–212, Cambridge, MA, August 2006.

[26] Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. A stochastic finite-state word-segmentation algorithm for Chinese. In *Proc. of the 32nd annual meeting on Association for Computational Linguistics*, pages 66–73, Morristown, NJ, 1994.

[27] Suzanne Topping. The secret life of Unicode. *IBM Developer Works*, http://www-128.ibm.com/developerworks/library/u-secret.html, May 2001.

[28] Jakob Voss. Measuring Wikipedia. In *Proc. of the 10th International Conference of the International Society for Scientometrics and Informetrics 2005*. International Society for Scientometrics and Informetrics, Stockholm, Germany, July 2005.

[29] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[30] Jinxi Xu, Alexander Fraser, and Ralph M. Weischedel. TREC 2001 cross-lingual retrieval at BBN. In *TREC '01: Text Retrieval Conference*, National Institute of Standards and Techonology, 2001.

[31] Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. Maximum entropy based restoration of Arabic diacritics. In *ACL '06: Proc. of the 41st Annual Meeting on Association for Computational Linguistics*, Sydney, Australia, 2006.

[32] J. Zobel and P. W. Dart. Phonetic string matching: Lessons from information retrieval. In H.-P. Frei, D. Harman, P. Schäble, and R. Wilkinson, editors, *Proc. of the 19th International Conference on Research and Development in Information Retrieval*, pages 166–172, Zurich, Switzerland, 1996.