March 26, 2013

Essential Elements of Multilingual Search

*Boosting Global Search Quality with the Rosette Linguistics Platform*
ABOUT BASIS TECHNOLOGY

Basis Technology provides software solutions for text analytics, information retrieval, digital forensics, and identity resolution in over forty languages. Our Rosette® linguistics platform is a widely used suite of interoperable components that power search, business intelligence, e-discovery, social media monitoring, financial compliance, and other enterprise applications. Our linguistics team is at the forefront of applied natural language processing using a combination of statistical modeling, expert rules, and corpus-derived data. Our forensics team pioneers better, faster, and cheaper techniques to extract forensic evidence, keeping government and law enforcement ahead of exponential growth of data storage volumes.

Software vendors, content providers, financial institutions, and government agencies worldwide rely on Basis Technology’s solutions for Unicode compliance, language identification, multilingual search, entity extraction, name indexing, and name translation. Our products and services are used by over 250 major firms, including Cisco, EMC, Exalead/Dassault Systems, Hewlett-Packard, Microsoft, Oracle, and Symantec. Our text analysis products are widely used in the U.S. defense and intelligence industry by such firms as CACI, Lockheed Martin, Northrop Grumman, SAIC, and SRI. We are the top provider of multilingual technology to web and e-commerce search engines, including Amazon.com, Bing, Google, and Yahoo!

**ABSTRACT**

If search is vital to the value proposition and competitive advantage of your business, then this whitepaper is a must-read. Companies in a wide range of markets thrive or languish depending on their ability to provide comprehensive and relevant search results and serve all users equally well in their preferred language.

The three essential elements of a commercial-caliber search solution are (1) search customizability and the speed-scalability-cost trio, (2) search result relevancy with an emphasis on “recall,” the percentage of relevant results found, and (3) reliability and support. This paper examines these three elements and how Basis Technology’s Rosette Linguistics Platform provides an ideal solution to the challenges they present.

**THE THREE ELEMENTS OF MULTILINGUAL SEARCH**

Revenue and business longevity are increasingly reliant on cost-effective, accurate, and scalable search in English and other languages. From e-commerce to e-discovery, from financial compliance to social media, nearly all information-driven industries rely in some way on the ability to enable users to find relevant information. Choose the right search solution and see revenue go up, customer satisfaction increase, and your business well-positioned for the future. Choose the wrong search solution and see costs constrict growth and low result relevancy chase away customers. Savvy businesses recognize that their customers, partners, suppliers, and employees are increasingly creating and consuming content in their native language.

For most enterprises, improving search quality means improving both recall, i.e., the percentage of relevant results found and, to a slightly lesser degree, precision, i.e., the ratio of good to bad results returned.

We’ll now examine each of these three elements of search in detail:

- **Customizability** of search is crucial. Tailoring a search solution to your particular use case allows you to create or enhance your value proposition. At the same time, speed, scale, and cost issues can quickly inhibit your business’ success. Search should be both fast and scalable so as not to impede your application’s overall performance, and per-query and per-document costs must be capped to maintain profitability.

- **Maximizing search recall and precision** is the 800-pound gorilla. Speed is important, but if your users aren’t finding the results they want, it doesn’t matter how fast they’re being delivered. For most enterprise applications, maximizing recall (while maintaining precision) is usually the greater concern. Good language support is imperative for high recall.

- **Reliability and technical support** allow you to focus on making your product great rather than on the nuances of multilingual search. Having expert resources at your fingertips is essential for the timely resolution of any issues that might arise.

**ELEMENT 1: CUSTOMIZABILITY AND THE SPEED-SCALE-COST TRIO**

The fundamental question your company must ask is whether customized search technology is necessary to achieve a competitive advantage in the context of your business model. If the answer is “yes,” then your search problem cannot be solved with a rigid plug-in solution, but rather will require custom development.

Fortunately, the highly-flexible Rosette Linguistics Platform was designed for a wide range of large-scale applications that need to analyze unstructured text from various sources, and so is a natural complement to both open source and proprietary search solutions. We offer easy
out-of-box integration with Apache Lucene/Solr, and are also in the process of developing an ElasticSearch connector. In addition, Basis Technology has over 15 years of experience supporting global customers whose needs require Rosette integration with homegrown or proprietary third-party search solutions. Nearly every major web and enterprise search engine since 1999, including Bing, Endeca, goo (Japanese), Google, Microsoft/FAST, and Yahoo!, has used Rosette. New generations of search-based applications for e-discovery, financial compliance, and other fields also use Rosette.

One API provides access to over 40 Asian, European, and Middle Eastern languages, so there's no need to find and integrate new solutions for every new language you encounter. Simply turn on the next language you need and keep using the same API. Whether your product is still up-and-coming or industry-leading, Rosette can be scaled to meet your needs such that you can start getting the benefit of enterprise-grade linguistic technology today. Additionally, Rosette's high-performance libraries have been tested to extreme scale, and are guaranteed not to be the bottleneck in your search stack.

Because search is such a broad and nuanced problem, there are all kinds of edge cases that are difficult to support using a single proprietary search solution. Such solutions are not easily customized without expensive consultation and modifications to core software that are tied to the vendor's product release cycle. For companies with unique use cases, the work required to tailor proprietary solutions to even partially meet their requirements is often prohibitively expensive.

Speed, scalability and cost are inextricably intertwined. Assuming that search engine load will increase over time, search functionality must be scalable in order to maintain fast query response times and ensure index updates do not significantly affect search performance. At the same time, pricing models of commercial search engines based on number of queries or number of documents indexed may eventually constrict growth. For these reasons, enterprise adoption of open source search solutions is becoming increasingly common. Since Lucene/Solr is currently the most mature and widely-used open source search platform, in the following section, we'll describe how Rosette and Lucene/Solr can be paired to provide a powerful, cost-effective, multilingual search solution.

**Rosette for Apache Lucene and Solr**

In recent years, Apache Lucene and Solr have become a viable alternative to commercial search technologies, due to their speed, scalability, and feature-richness, coupled with the transparency and extensibility intrinsic to being open source, which lowers implementation and maintenance costs.

**Apache Lucene** is a pure-Java high-performance indexing and search library that allows fine-grained control of machine functions and independence from higher-level protocols. Various open source ports of Lucene to other languages including C++, .NET, C, Python, and Ruby are available. If your application has very specific requirements that necessitate lower-level control, it may make sense to use Lucene directly, rather than via Solr.

**Apache Solr** is a highly-scalable, easily-configurable, production-ready enterprise search platform built on top of Lucene with REST-like HTTP/XML and JSON APIs that make it readily accessible from any programming language. It features hit highlighting, faceting, rich document handling, database integration, and near real-time indexing. Solr runs on a lightweight platform-independent Java server, making it easy to leverage advances in hardware and operating systems while minimizing ongoing development costs.
Today, thousands of organizations, including Twitter, LinkedIn, Netflix, CNET, Apple, Wikipedia, and Zappos, leverage Lucene and Solr to power a wide range of search applications. In many cases, Lucene/Solr solutions regularly index and search millions to billions of documents with sub-second response time, making Lucene/Solr-based enterprise search solutions among the most robust available.

Out of the box, Lucene and Solr offer some language support; however, the level provided is insufficient to raise multilingual search quality to the standard demanded by commercial-caliber applications. This is where Rosette comes in: Rosette's pre-built JAR files, which integrate seamlessly with either pure Lucene or Lucene/Solr, provide sophisticated natural language processing technology as well as comprehensive dictionary data to fill in the gaps.

More specifically, Lucene/Solr provides stemmers for many languages, tokenization for Chinese, Japanese, and Korean via the CJK Tokenizer, and a mechanism for handling compound words in Germanic and Scandinavian languages. Rosette provides lemmatization in over 25 Asian, European, and Middle Eastern languages, which, as we'll discuss in the next section, yields higher precision and recall than stemming. Additionally, Rosette's tokenizer dramatically improves upon the provided CJK Tokenizer, by using full morphological analysis rather than a simplistic bigram approach, resulting not only in higher precision and recall but also a smaller index. While Solr incorporates a decompounding algorithm, it lacks the dictionary data required to utilize its functionality. Rosette's decompounding for German, Dutch, Danish, Norwegian, and Swedish, as well as Japanese and Korean, is complete with comprehensive dictionary data. For these and various other reasons, Rosette is required to leverage the full power of Lucene/Solr in multilingual environments.

**ELEMENT 2: SEARCH RESULT RELEVANCY**

We can view all types of search, whether the context is a shopper using an e-commerce site, a banker verifying a wire transfer request against a list of money launderers, or a home buyer scanning local real estate listings, as information retrieval problems. In the context of information retrieval, precision and recall measure the efficacy of a system. Thus, higher precision and recall indicate higher-quality search. In practice, increasing recall often leads to a drop in precision, and vice versa, so the challenge lies in maximizing the value of both metrics concurrently.

**Precision, Recall, and F-Score in a Nutshell**

In the context of search, precision is the fraction of retrieved results that are relevant, and recall is the fraction of relevant results that are retrieved. F-score is a measure that combines both precision and recall, and hence, is a good indicator of a search solution's relevancy.

To clarify, let's look at a simple example. Suppose you are searching for red balls from a box that contains 7 red balls and 8 green balls. Blindfolded, you pull out 8 balls, 4 red and 4 green.

Precision is the number of correct results over the total number of results retrieved. So, in this case, since you found 4 red balls out of a total of 8 balls retrieved, precision is 4/8 or 50%.

Recall is the number of correct items you found over the total number of correct items. So, in this case, since you found 4 out of 7 total red balls, recall is 4/7 or 57%.

F-score is calculated by taking the harmonic mean of precision and recall. In our example, $F = \frac{(2 \times \text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} = \frac{8}{15}$ or 53%
Sophisticated natural language processing technology is required to maximize precision and recall in full-text search. In particular, lemmatization is crucial for highly-inflected languages. In the following sections, we’ll examine why lemmatization matters, as well as the importance of language-specific analysis across three language categories notoriously difficult to process: Chinese/Japanese/Korean, Germanic and Scandinavian, and Arabic. We’ll also take a look at how entity extraction and name technology can improve result relevancy.

**Improving Precision and Recall with Lemmatization**

In linguistic terms, many European languages are highly inflected. This means that words are modified based on tense, gender, case, quantity, aspect, and other grammatical categories. Even English, which is considered weakly inflected, contains conjugated verbs and a number of irregular nouns. Some European languages also prominently feature declension, or the inflection of nouns, adjectives or pronouns. In short, it is common for the forms of words to change based on how they are used.

To address this challenge, words must be normalized at both index and query time, to ensure that a query for one form of a word matches all occurrences of the word, regardless of form. There are two main approaches that a search engine can use to normalize word variations:

- **Stemming** — a simple rules-based approach that removes characters from the end of a word in order to reduce it to its root form.
- **Lemmatization** — the identification of the dictionary form of a word based on its context.

For European languages, lemmatization is preferable to a simplistic stemming approach because it improves both precision and recall. For example, a user searching for “President Obama speaking on healthcare” would likely also want results containing “President Obama spoke on healthcare” and “President Obama has spoken on healthcare.” By lemmatizing “speaking,” “spoke,” and “has spoken” to the lemma “speak,” at both indexing and query time, you are able to compare apples to apples and hence return all relevant results to the user. In contrast, using stemming, there is no way to derive “speak” from “spoke.” Additionally, stemming often yields results that are either non-dictionary words or dictionary words that differ in meaning from the input. For example, the stem of “decoding,” “decoder,” and “decodes” is “decod,” and stemming turns “several” into “sever,” and both “arsenic” and “arsenal” into “arsen.”

Over many years, enterprise search users have confirmed that lemmatization improves search quality in a wide range of scenarios, while stemming has limited utility. Below is a list of sample words that demonstrate some of the problems associated with stemming algorithms and how they are resolved using lemmatization.

**Stemming vs. Lemmatization Examples**

<table>
<thead>
<tr>
<th>Search Query</th>
<th>Traditional Stemming</th>
<th>Lemmatization using Rosette</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>animals</td>
<td>anim</td>
<td>animal</td>
<td>Two unrelated words may share a common stem.</td>
</tr>
<tr>
<td>animated</td>
<td>anim</td>
<td>animate</td>
<td></td>
</tr>
<tr>
<td>arsenal</td>
<td>arsen</td>
<td>arsenal</td>
<td></td>
</tr>
<tr>
<td>arsenic</td>
<td>arsen</td>
<td>arsenic</td>
<td></td>
</tr>
<tr>
<td>several</td>
<td>sever</td>
<td>several</td>
<td>Stemming may have unintended consequences.</td>
</tr>
<tr>
<td>organization</td>
<td>organ</td>
<td>organization</td>
<td></td>
</tr>
<tr>
<td>children</td>
<td>children</td>
<td>child</td>
<td>Irregular verbs and nouns stump the stemmer.</td>
</tr>
<tr>
<td>spoke</td>
<td>spoke</td>
<td>speak (used as past-tense verb) speak (used as noun)</td>
<td></td>
</tr>
</tbody>
</table>
Improving Precision in Chinese, Japanese, and Korean

Asian languages like Chinese, Japanese, and Korean are fundamentally more difficult to process than European languages because, in these languages, words are not space-delineated. In Chinese and Japanese, there are no spaces between words; in Korean, spaces delineate words inconsistently and with some variation among writers.

For a search engine to build an index, it requires discrete words as input. There are two main approaches commonly used to segment Asian text into a series of discrete words:

- **N-gram**, or chopping text into segments n-characters long, where n is typically 2 to 3.
- **Morphological analysis**, or utilizing dictionaries and language-specific rule-sets with knowledge of language features to segment text more accurately.

### N-gram vs. Morphological Analysis

While the n-gram approach requires less back-end programming, it yields far less satisfying results for end-users, because although recall will be high, precision will be low and indexes will be significantly bloated, negatively impacting performance.

Here is a Japanese example: 東京都の観光地 (translation: sightseeing spots in Tokyo)

<table>
<thead>
<tr>
<th>Bigram</th>
<th>English translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>東京</td>
<td>Tokyo</td>
</tr>
<tr>
<td>京都</td>
<td>Kyoto</td>
</tr>
<tr>
<td>都の</td>
<td>capital's</td>
</tr>
<tr>
<td>の観</td>
<td>(not a word)</td>
</tr>
<tr>
<td>観光</td>
<td>sightseeing</td>
</tr>
<tr>
<td>光地</td>
<td>(not a word)</td>
</tr>
</tbody>
</table>

Using the n-gram method with n = 2 and this seven-character phrase as input produces all possible two-character substrings, for a total of six tokens to add to the index, including both one dictionary word not present in the original phrase (Kyoto), and two non-word strings. As demonstrated by this example, because Japanese words are typically only a few characters in length, the number of “false words” created by n-gram segmentation is relatively high, and hence will result in many false matches, decreasing search precision.

A better choice is tokenization based on morphological analysis, which breaks streams of text into the actual words that compose it. Using morphological analysis to tokenize the same example phrase produces at most three tokens to index, and possibly only two if the possessive marker is treated as a “stop word.”

---

1. In linguistics, morphological analysis is the identification and description of a language's morphemes, or smallest units of meaning.
2. A “stop word” is a word that occurs so frequently in a language that it conveys little useful information in the context of search. For this reason, some search engines remove stop words prior to further text processing to improve result relevancy.
### Improving Recall with Character Normalization

Chinese, Japanese, and Korean have additional character normalization needs beyond making sure ASCII characters are all uppercase or all lowercase. In digital files, these languages also use a full-width variant of ASCII letters and punctuation that needs to be normalized to the half-width ASCII form.

\[
\text{ASCII\text{words}}\%\ $ &\rightarrow \text{ASCII\text{words}}\%\$&
\]

In the case of Japanese, a half-width version of Japanese katakana characters also needs to be normalized to the usual full-width katakana. The half-width version was invented in the early days of computer processing to reduce data size, but it is still found in webpages and other documents.

\[
\text{katakana} \rightarrow \text{katakana}
\]

### Improving Recall with Pan-Chinese Search

Chinese search gains an enormous boost in recall if both the simplified and traditional Chinese scripts are searched. Mainland China and Singapore use simplified Chinese, whereas Taiwan and Hong Kong use traditional Chinese. Chinese speakers expect one query in one script to find results from both scripts.

Pan-Chinese search — searching documents in both Chinese scripts — is accomplished by converting all the text of one script into the other at query and index time.

The differences between the two scripts fall into three categories:

<table>
<thead>
<tr>
<th>Category</th>
<th>Simplified Chinese</th>
<th>Traditional Chinese</th>
<th>English Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same character used in both scripts — this is true where the character was simple to start with</td>
<td>大</td>
<td>大</td>
<td>big</td>
</tr>
<tr>
<td>Two characters with same pronunciation (but different base meaning) — collapsed to one character in simplified Chinese</td>
<td>头发</td>
<td>頭髮</td>
<td>hair emit</td>
</tr>
<tr>
<td>Different vocabulary (analogous to American and British English differences such as “truck” vs. “lorry”) — occurs in mostly modern words</td>
<td>计算机</td>
<td>電腦</td>
<td>computer</td>
</tr>
</tbody>
</table>

In the first case, the change is trivial, and amounts to making sure all the text is in the same encoding; however, the second and third categories require dictionary data to ensure that the conversion is context-sensitive, so that the correct traditional character is chosen.

### Improving Recall in Germanic and Scandinavian Languages

Linguistic processing required by German, Dutch, Scandinavian languages (Danish, Norwegian, Swedish, and Finnish), and Korean is very similar in that these languages freely use compound words which frequently need to be broken up to increase recall.
Take these two German compound words:

<table>
<thead>
<tr>
<th>German compound</th>
<th>German decompounded</th>
<th>English translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jugendarbeitslosigkeit</td>
<td>Jugend + arbeitslosigkeit</td>
<td>Youth unemployment (Youth + unemployment)</td>
</tr>
<tr>
<td>Samstagnmorgen</td>
<td>Samstag + morgen</td>
<td>Saturday morning (Saturday + morning)</td>
</tr>
</tbody>
</table>

It’s reasonable to imagine that a person looking for information about youth unemployment in German would also welcome search results that included “youth” and “unemployment” as separate words. Similarly, the concepts of “Saturday” and “morning” are very likely to appear as a compound word or as separate words in related documents.

Decompounding increases search recall with minimal impact on precision, but again, requires dictionary data to do properly.

Additionally, to maximize recall, character normalization for plurals is required. For example, “Garten” and its plural form “Gärten” should be considered a match.

**Improving Recall in Arabic**

Arabic is one language that especially suffers from low recall if a search engine does not perform Arabic-specific linguistic processing, starting from the basic character normalization up to stemming of proper nouns and lemmatization of common nouns. Arabic attaches so many affixes — to the beginning, end and middle of words — that without lemmatizing before searching, many relevant results are never found. One root could form the basis for up to fifteen different verbs!

**Improving Precision and Recall with Character Normalization**

In English, search engines perform some basic normalization such as lowercasing all the words. In Arabic, normalization is much more complex and will significantly improve both precision and recall. Types of character normalization required include:

- Words with additional vocalization marks such as: سياسى vs. سيابى
- Words containing certain letters with dots added or removed such as: فترة vs. فریه
- Words — including ambiguous cases — containing certain letters with symbols added or removed such as:
  - اعتقاد vs. آعتماد
  - داوود vs. داوود
  - (أئم or إئم or آئم) vs. (أئم or إئم or آئم)

**Improving Recall with Lemmatization and Stemming**

In Arabic as other languages, lemmatization increases the recall of search results (cf. “Improving Precision and Recall with Lemmatization”), but with a twist. In Arabic, lemmatization is only applicable to verbs and common nouns (e.g., apple, book, table) and is not applicable to proper nouns (e.g., names of people such as Abul-qassem El-Chabby, and Baqah Al-Sharqiyyah).
Proper nouns require stemming to remove prepositions and conjunctions that are often attached to them. Searches for names of people, places, and organizations are seriously hampered without stemming. For example, these phrases in English appear as one word in Arabic: “for Othman,” “with Othman,” “as Othman,” and “and Othman.” Thus a search for just “Othman” would not find the above variations without stemming.

Additionally, since Arabic names are frequently the same words as common nouns, part-of-speech tagging becomes critical to differentiating between nouns and proper nouns, particularly when a word’s part-of-speech varies depending on where it appears in a sentence.

**Search Recall and Precision Enhancers for Many Languages**

For English and most European languages, better recall and precision means:

- Adding lemmas (the dictionary form of a word) to the search index to increase recall, while minimizing the negative impact on precision (cf. previous section “Improving Precision and Recall with Lemmatization”)
- Boosting the ranking of relevant documents on the results page through the use of document metadata such as ensembles.
- Increasing recall through comprehensive name search that goes beyond the “Did you mean?” functionality of most search engines.

**Boosting Relevant Results**

Once the search engine is finding more good results (better recall) and returning fewer bad results (better precision), the good results need to be at the top of the search results.

Frequently, the most critical words in a search have to do with ensembles — proper nouns such as names of people, places, and organizations. But when is “Christian” a religious affiliation and when is it the name of the person, “Christian Dior”? A good entity extractor will know. At indexing time, entities can be added to a document record’s metadata to boost the rank of documents that have entities matching those in the query.

**Comprehensive Name Search to Improve Recall**

The “Did you mean?” function of most modern search engines will find actress “Cate Blanchett” even if the user types “Kate Blanchett.” However, for less famous people, and to cover a gamut of name variations, a true name matching function is needed to handle “Chuck Berry” vs. “Charles Berry”; “John Kearns” vs. “Jon Cairns” or even “Baqah Al-Sharqiyyah” and “باقية الشرقية”.

**The Rosette Option for Language Support**

Rosette is a software development kit (SDK) designed for a wide range of large-scale applications that need to identify, classify, analyze, index, and search unstructured text from various sources, and its linguistic analysis capabilities are widely used by search engines to both increase precision and recall. It uses a combination of statistical models, dictionary data, and sophisticated computational linguistics to parse digital text in English and over 25 major European, Asian, and Middle Eastern languages. Years of development and linguistic analysis have gone into the development of Rosette to satisfy the most demanding customers, both for quality, robustness, and speed.

Rosette is a cross-platform SDK available for Windows and Unix, and offers all its capabilities via a single API, in C, C++, Java, or .NET.
Rosette Capabilities

- **Language Identification** — identification of the primary language of a document—or language regions within a multilingual document—and the file’s encoding in 55 languages and 45 encodings.

- **Unicode Conversion** — converts documents in legacy encodings to Unicode

- **Character Normalization** — normalizes characters to a single representation (e.g. character + diacritic to single character with diacritics, half/full-size variants of ASCII characters in Asian languages, and several language-specific normalizations.)

- **Linguistic Analysis** — lemmatization, tokenization, part-of-speech tagging, decompounding, and more in over 25 languages.

- **Entity Extraction** — extracts entities such as people, places, and organizations in 15 languages via statistical models with customizable user-defined entities via regular expressions and entity databases

- **Name Matching** — returns matching names despite spelling variations, initials, nicknames, missing name components, missing spaces, out-of-order name components, the same name written in different languages, and more—supported in multiple languages including Arabic, Chinese, English, Korean, and Persian.

- **Name Translation** — translates names from non-Latin script languages to English and standardizes already translated names—supported in multiple languages including Arabic, Chinese, English, Korean, Russian, and Persian.

**ELEMENT 3: RELIABILITY AND SUPPORT**

Even in a technologically sophisticated company, the core business logic will be the main focus of all business operations, so technical support as a safety net is essential. Just as firms rely on calling the vendor when the copier is acting up, a few judicious pieces of advice from the search or linguistics experts to settle a nettlesome problem lends assurance to both engineers and upper management.

On the language-support side, the wide language coverage of Basis Technology’s Rosette — over 25 languages covering Asia, Europe, and the Middle East — ensure there will be one point of contact to address questions about any languages, instead of a multitude of single-language vendors with varying support contracts. Rosette developers are on the front lines, providing high-quality support to customer inquiries.

Additionally as a single platform, Rosette’s benchmarks for speed and accuracy are readily available, and implementing one or over 25 languages is the same amount of work.

**NEXT STEPS**