2005

Cross-Language Retrieval for Arabic Texts: The Creation of an English-Arabic Cross-Language Information Retrieval Environment

Robert N Oddy, *Syracuse University*
Anne R. Diekema, *Syracuse University*
Jean Hannouche, *Syracuse University*
Grant Ingersoll, *Syracuse University*
Elizabeth D. Liddy, *Syracuse University*

FINAL TECHNICAL REPORT

Cross-Language Retrieval for Arabic Texts: The Creation of an English-Arabic Cross-Language Information Retrieval Environment

Anne R. Diekema, Jean Hannouche, Grant Ingersoll, Robert N. Oddy, and Elizabeth D. Liddy

CNLP Technical Report #01202005-1

Cross-Language Retrieval for Arabic Texts: The Creation of an English-Arabic Cross-Language Information Retrieval Environment

Anne R. Diekema, Jean Hannouche, Grant Ingersoll, Robert N. Oddy, and Elizabeth D. Liddy

Center for Natural Language Processing
School of Information Studies
Syracuse University
Syracuse, NY 1244-4500
diekemar@syr.edu

Abstract
An English-Arabic Cross-Language Information Retrieval Environment was created in which the user can query an Arabic database in English and retrieve a set of relevant Arabic documents. The retrieved Arabic documents will be automatically translated into English to facilitate readability by the English language user. Proper names of people, places, and organizations are extracted from the retrieved documents and transliterated from Arabic into English. They are presented to the user and serve to provide a brief summarization of the retrieved document. Another feature of the AIR design is the user's ability to group searches and search results into what we call Topics which persist between sessions and can be managed by the individual user.

The guiding principle in the AIR system is to get away from the English query as soon as possible and rely on relevance feedback to refine the Arabic version of the query, thereby providing the user with helpful information as quickly as possible. High precision query translation comes from the combination of different lexical resources to improve translation probabilities of initial query terms, and also to provide high-quality data for the interactive sense-disambiguation tool. The lexical combinatory resource includes machine-readable dictionaries, ontologies, machine translation lexicons, encyclopedias, and comparable corpora.

Keywords
cross-language information retrieval, CLIR, translation, transliteration, Arabic language, evaluation, lexical resource creation, natural language processing

TABLE OF CONTENTS

FIGURES AND TABLES

1. Introduction

The project created an English-Arabic Cross-Language Information Retrieval Environment (AIR), in which the retrieval system enables a user to query a database of Arabic documents in English and retrieve a set of relevant Arabic documents. The retrieved Arabic documents are automatically translated into English to facilitate readability by the English language user. Proper names of people, places, and organizations are extracted from the retrieved documents and transliterated from Arabic into English. They are presented to the user and serve to provide a brief summarization of the retrieved documents.

The guiding principle in the AIR system is to get away from the English query as soon as possible and rely on relevance feedback to refine the Arabic version of the query, thereby providing the user with helpful information as quickly as possible. The system focuses on high precision in the first few (e.g. 10) documents, then uses pseudo-relevance feedback to pick up good Arabic query terms and weights. This step is included to boost query translation. An additional user feedback stage is also provided, to allow the user to manipulate the original query based on the retrieved documents.

The high precision comes from the combination of different lexical resources to improve translation probabilities of initial query terms, and also to provide high-quality data for the interactive sense-disambiguation tool. The lexical combinatory resource includes machine-readable dictionaries, ontologies, machine translation lexicons, encyclopedias, and comparable corpora. While resources are valuable individually, automatic intelligent combination into a single resource has shown value that is greater than the sum of its parts. A lexical combinatory resource can provide better coverage, more reliable translation probability information, and additional information leveraged through the process of lexical triangulation. Proper names that are not listed in the query translation resource are transliterated into Arabic to facilitate matching of people, places, and organizations across English and Arabic.

Another feature of the AIR design is the user's ability to group searches and search results into what we call Topics. The searches may have been initiated with different English queries, and may have been conducted on different databases, but then relevance feedback for a new query operates across all the queries within a topic. Topics are owned by individual users and persist between sessions, and can be shared with other analysts.

2. Motivation and methods

2.1 Arabic

An estimated 300 million people around the world speak Arabic (Aljlayl and Frieder, 2001), making it the world's eighth most common language. It is also one of the six official languages adopted by the United Nations and the official language of Algeria, Bahrain, Egypt, Iraq, Jordan, Kuwait, Lebanon, Libya, Morocco, Oman, Qatar, Saudi Arabia, Sudan, Syria, Tunisia, United Arab Emirates, and Yemen. While spoken Arabic

varies between countries, classical written Arabic (the language of the Koran) is shared among all. Regional variations in spelling do exist.

The Arabic alphabet has 28 letters, mostly consonants. Diacritics above the letters indicate the presence of vowels. However, these markings are often left out in printed text (non-vocalized Arabic). The shape of most of the characters depends on their position in a word and the adjacent characters. Arabic is written from right to left. In Malta, the Arabic language is written using the Latin alphabet.

The morphological variation of Arabic words is substantial, especially when compared to English. Starting from a closed set of roots (±10,000), one can create all necessary vocabulary by adding prefixes, infixes, and suffixes, replacing vowels and omitting letters (Darwish and Oard, 2002_1). Roots vary in length from 3 to 5 characters, but most roots are made up of three consonants. A distributional analysis of Arabic newspaper text by Larkey, Ballesteros, and Connell (2002) showed that Arabic had more unique words than English text samples of identical size.

According to Xu, Fraser, and Weischedel (2002) Arabic is a challenging language for information retrieval because:

1) orthographic variations are prevalent in Arabic; certain combinations of characters can be written in different ways, regional differences in spelling may exist
2) Arabic has a very complex morphology
3) broken plurals (analogous to irregular nouns in English) are common
4) Arabic words are often ambiguous due to the tri-literal root system
5) short vowels are omitted in written Arabic making it more ambiguous and harder to match with translation resources that may be vocalized
6) synonyms are widespread.

2.2 Representing Arabic text

While the English character set can be encoded using ASCII, this encoding scheme is insufficient to represent the complex character set of Arabic. Before a suitable encoding set was available, Arabic documents on the web were often displayed as an image rather than text. Several encoding schemes exist for Arabic i.e. UTF-8, CP1256 (windows), ISO 8859-6, and ASMO 708. Most Arabic IR researchers use UTF-8 which is an encoding scheme that is part of UNICODE. The benefit of using UTF-8 is that the Java programming language uses UTF-8 internally to represent character sets and that Java-based retrieval engine Lucene is thus UTF-8 compatible. Another solution to the encoding problem is to transliterate all Arabic into a Latin-based script. This approach was used by two TREC teams (Darwish et al., 2001 and Savoy and Rasolofo, 2002). Darwish used his own scheme while the Université de Neuchâtel used the system that is used in Malta where a Latin alphabet version of Arabic is used. (Savoy and Rasolofo, 2002)

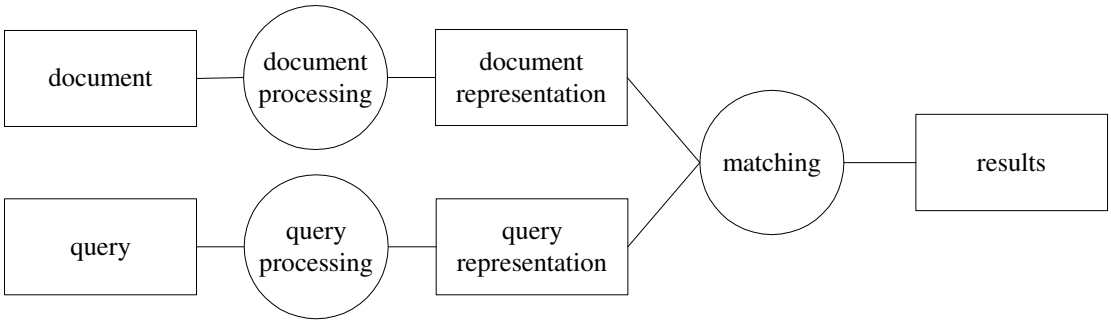

Figure 1. General Information Retrieval process.

2.3 The CLIR process

Generally speaking, Cross-Language Information Retrieval (CLIR) is not that different from monolingual IR. Documents are processed and stored in the system, users' queries are processed, and then matched (see Figure 1). The main difference is, of course, that we are trying to match across languages and some sort of translation is required to accomplish this. As we will see below, this complication requires additional steps in query processing (since we plan to use query translation), matching, as well as the presentation of results. Another important difference is that we are dealing with text encoded in different coding schemes than those we are used to.

When we examine the different IR processes in detail (circles in Figure 1), the extra requirements due to multi-linguality become apparent. The components in the general CLIR process such as document processing, query processing, and matching will be discussed in the paragraphs below.

2.4 Document processing

Document processing involves going through all the documents in the collection to extract indexable terms and to calculate document specific and collection specific statistics in the process. Documents first need to be pre-processed, tokenized, and normalized before terms can be extracted (see Figure 2). Ultimately the terms and their weights are stored in a searchable index.

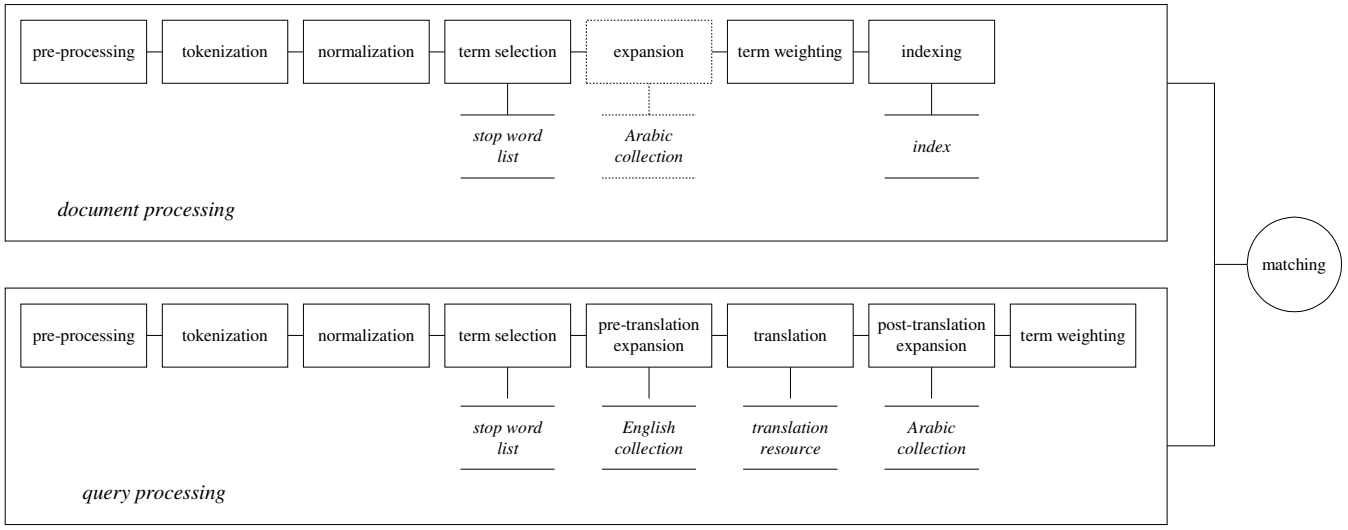

Figure 2. The CLIR process.

2.4.1 Document pre-processing

During document pre-processing, the original documents are converted into a format that the CLIR system can use. For example, the TREC collection usually comes in a number of zipped files, each of which contain a number of documents. These files need to be unzipped and possibly segmented into individual documents, or passages.

2.4.2 Tokenization

During the tokenization phase, the document text is split into individual tokens. Plain Arabic tokenization is similar to English in that you can split a text using the white-space character and punctuation. Larkey and Connell (2001) adapted their regular tokenizer by adding five additional Arabic punctuation characters.

2.4.3 Orthographic and term-level normalization

Normalization is needed in information retrieval to facilitate consistent matching between documents and queries. At this stage the (Arabic) text may be mapped into a single encoding standard, punctuation may be removed, diacritics normalized (see section 4.3.1), and tokens may be normalized through some level of morphological analysis (see section 4.3.2).

2.4.3.1 Orthographic normalization

There is a great deal of orthographic variation in Arabic because many diacritics are optional. To facilitate matching it is best to remove these diacritics even though this may increase ambiguity. In addition, some Arabic characters can be written multiple ways depending on their position in a word. These characters have several Unicode representations and to facilitate matching it is best to map various forms into a single

normalized form. For example the letter Kaf in isolated form (ﮓ), looks different from the letter Kaf in final form (ﮓ), or initial form (ﮕ), or medial form (ﮕ). Several papers provided details as to what different Arabic CLIR research teams normalized (Larkey, Ballesteros, and Connell, 2002), (Chen and Gey, 2001), (Larkey and Connell, 2001), (Frasier, Xu, and Weischedel, 2002), (Tomlinson, 2002), (McNamee, Piatko, and Mayfield, 2002), (Darwish and Oard, 2002_2), (Larkey et al., 2002). For example Darwish and Oard (2002_2) removed all diacritics and kashidas and normalized the letters Ya (ي) and Alif Maqsoura (ى) to Ya (ي) and all the variants of Alif (ا) and Hamza (ء) to Alif (ا). Larkey and Connell (2001) removed vowel-diacritics and non letters and replaced an initial إ or أ with bare Alif (ا), replaced آ with ا, replaced the sequence ىء with ئ, replaced the common suffix ى with ي, and finally replaced the common suffix ة with ه.

2.4.3.2 Morphological analysis

Since Arabic has a rich morphology it is better to use some sort of normalization of a term instead of using the surface form.

According to Levow, Oard and Resnik (under review), determining a root form of a term is very complex since multiple roots can generate identical tokens. As a solution to this problem they suggest that one:

1) use corpus statistics to determine possible roots and the most probable choices.
2) apply rule-based techniques to remove common prefixes and suffixes
3) group terms into classes using corpus based clustering

2.4.3.2.1 Stemming

Stemming is used in retrieval to handle the vocabulary mismatch problem by conflating related words (i.e. biking, bike, and biker) into groups (i.e. bike). Strong stemmers create larger groups with the danger of conflating unrelated terms, while weak stemmers might fail to group related terms (Larkey, Ballesteros, and Connell, 2002). A repartitioning technique based on term co-occurrence statistics was applied unsuccessfully to correct stemming errors by Larkey, Ballesteros, and Connell (2002). The same researchers also tried to avoid erroneous stemming of Named Entities by creating a list of stems for the stemmer to ignore. Unfortunately retrieval performance was actually better without the list. It is worth noting that Savoy and Rasolofo (2002) do not stem before splitting words into *n*-grams but rather removes very frequent tri-grams later.

Stemming in Arabic can be achieved through the use of: 1) manually constructed dictionaries (cannot stem terms that are not listed), 2) algorithmic light stemmers, 3) morphological analyzers to find root forms, or 4) statistical techniques to cluster similar terms. The easiest and most successful way to stem Arabic appears to be the so-called “light stemming”. Light stemming is actually a combination of orthographic normalization and prefix and suffix removal. It is called “light” because the resulting stems are not always identical to stems produced by full linguistic analysis (Levow, Oard, and Resnik, under review).

Numerous papers provide listings of the prefixes and suffixes that their light stemmers remove (Chowdhury et al., 2002), (Chen and Gey, 2002), (Larkey et al., 2002), (Savoy and Rasolofo, 2002). We can easily re-create these stemmers for our system if so desired. A data driven approach to stemming was used by Chen and Gey (2001) who only removed suffixes after checking whether the term without the suffix was present in the collection both with and without the suffix. Others used a machine translation (MT) system to create a stemmer (Chen and Gey, 2002) by conflating all Arabic terms that translated into the same English term.

2.4.4 Term selection

There are several options when selecting what to index. Several TREC[1] research groups doing English-Arabic CLIR experimented using different surface forms (roots, stems, *n*-grams), or combinations thereof (Gey and Oard, 2001, Oard and Gey, 2002). Stems tend to outperform roots, which tend to outperform using words. Naturally there are terms that are not worthwhile to select at all and they are commonly listed in a stopword list.

2.4.4.1 Stoplist

Gey and Oard (2001) report that using a stoplist improved retrieval performance slightly but consistently over all queries. Using a stoplist has the additional benefit of reducing the index size and speeding up search time. To reduce the size of the Arabic stopword list, removal of stop terms is best done after stemming or morphological analysis.

A stopword list of 1,131 was created by Chen and Gey (2001) who used machine translation to translate their English stoplist and augmented this list with words from the Arabic-English glossary from Elementary Modern Standard Arabic. A similar approach was used by Oard and Gey (2002) and Chen and Gey (2002) who added all Arabic terms that translated into an English stopword to their 3,447-entry and 2,942 stemmed stoplists respectively. Other research teams used existing stoplists such as Darwish et al. (2001) who used the Sebawai (see section 8.7 on resources below) which contains 130 particles and pronouns, and Larkey and Connell (2001) who used the stoplist that comes with the Khoja morphological analyzer.

2.4.4.2 Words

[1] The Text REtrieval Conference (TREC) is a yearly event in which information retrieval systems run identical retrieval experiments for a grand scale comparative evaluation. The goals of TREC are: 1) to encourage research in text retrieval based on large test collections; 2) to increase communication among industry, academia and government by creating an open forum for the exchange of research ideas; 3) to speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems; and 4) to increase the availability of appropriate evaluation techniques for use by industry and academia, including development of new evaluation techniques more applicable to current systems (Voorhees and Harman, 1999).

Using the surface forms of Arabic forms does not do well in retrieval since the language is so morphologically complex. Singular word forms cannot match plural words for example.

2.4.4.3 Roots

Darwish and Oard (2002_1) note that the Arabic language has a closed set of approximately 10,000 roots. All Arabic words are derived from these roots, and to make matters more complex, multiple roots could create the same word. For verbs, the root is the base form and usually contains three or four consonants although five-letter roots exist as well (Chowdhury et al., 2002).

2.4.4.4 Stems

To get a stem in Arabic, one typically removes prefixes and suffices. This does not necessarily get you a root though because the infixes are left untouched.

2.4.4.5 *n*-grams

N-grams are created by moving a window of a certain size (*n*) across document text and indexing each string that appears in the window. Some approaches record *n*-grams across word (or stem) boundaries and others do not (Chen and Gey, 2002). The advantages of using *n*-grams is that they allow partial matches in case of spelling variations. For the same reason *n*-grams are successful in retrieval of OCR-ed Arabic text (Darwish and Oard, 2002_1). A 17% relative improvement in mean average precision was reported by (Mayfield et al., 2001) when using *n*-grams. Xu, Fraser, and Weischedel (2002) experimented with using *n*-grams based on words and *n*-grams based on stems and found that the latter approach did better from a retrieval performance perspective. They speculated that unstemmed words share too many prefixes and suffixes between them, creating many false matches. This problem may be solved by simply removing very frequent *n*-grams as was done by Savoy and Rasolofo (2002). Xu, Fraser, and Weischedel (2002) also tried *n*-grams of varying length (2, 3, 4) and found that the tri-grams did best. Savoy and Rasolofo (2002) found that tri-grams without stemming did as well as retrieval with word stemming. Contrary to this fact, Johns Hopkins/APL found that four-grams worked best (Mayfield et al., 2001). In the second TREC Arabic track, APL's official run was a combination of 3-, 4-, and 5-gram runs (Oard and Gey, 2002).

2.4.5 Document expansion

Darwish and Oard (2002_2) tried document expansion. They identified the 20 most descriptive terms from each document and used these terms to retrieve additional documents from the collection. The terms from the top-10 documents were then added to the original document. The document expansion technique is similar to query expansion but is carried out before indexing, on a document by document basis. This approach did not work very well and should probably not be pursued.

2.4.6 Term weighting

Typically, the similarity between documents and queries is determined by counting the occurrence of query terms in individual documents and the occurrence of these terms in the document collection as a whole.[2] These counts are based on the assumption that: 1) the more often a term occurs in a document, the more likely it is to convey the subject of that document, and; 2) terms that occur in only a few documents are often more valuable than terms that occur in many documents. Valuable terms have the ability to differentiate between documents and without them it would be quite difficult to make the distinction between relevant and non-relevant documents. Because some documents are long and are more likely to contain multiple occurrences of terms than short documents, document length is also taken into account. The assumption is that a term is a better subject indicator for a short document if it appears multiple times than it would be if it appeared an equal number of times in a long document. Also, terms have a greater chance of co-occurring in longer documents and might cause spurious correlations.

The document term counts (*term frequency*), the collection term counts (used in the *inverse document frequency*), and document length are combined in a so-called term weighting function. This weight is commonly referred to as a *tf·idf* weight because it is a multiplication of the term frequency (divided by the document length) and the inverse document frequency. [3] A weight can be calculated in this manner for each term in a document. The similarity score for a document can be calculated by summing all term weights. Document terms that do not occur in the query get the weight zero. Hence documents that do not contain any query terms will automatically get a score of zero. Retrieval systems differ in the formulas used for counting frequencies, and in the algorithms they use to calculate the similarity between documents and queries. Once a score has been calculated for each document, the documents are presented to the user in ranked order, with the most relevant document (the document with the highest score) in the first position.

2.4.7 Indexing

All TREC2001 CLIR Arabic track participants used some form of bag-of-word indexing (Gey and Oard, 2001). Indexing of either 3-grams, 4-grams, or stemmed terms appears to be most successful. Some teams tried combinations of these and these methods outperformed uniform indexing (Darwish et al., 2001). Darwish and Oard (2002_2) reported an unusual means of indexing called balanced translation indexing where each Arabic term in the document is replaced with its English top 5 most likely translations in the index during the indexing process.

---

[2] Once *words* are used in query or document representations they are typically referred to as *terms*.

[3] $w_{ij} = \frac{freq_{ij}}{length_j} * \log\left(\frac{N}{n_i}\right)$ where,

w_{ij} = weight of term i in document j, $freq_{ij}$ = frequency of term i in document j, $length_j$ = length of document j, N = the number of documents in the collection, n_i = the number of documents with term i.

2.5 Query processing

Query processing involves taking the query and extracting terms to match against the document index (see figure 2).

2.5.1 Query pre-processing

Query pre-processing usually requires minimal work except for TREC evaluations when queries are fed to the system in batches. Each TREC topic (as questions are called in TREC) has a title, description, and narrative section, each separated by SGML tags. Queries are created based on these topics. Certain TREC topic specific phrases (i.e. *relevant documents describe*) need to be removed from the queries as well. Queries may also need to be converted to an encoding that matches the translation resource being used in the system.

2.5.2 Tokenization, normalization, and term selection

Query processing steps such as normalization, and term selection are identical to document processing. It is important to use the exact same approach in both query and document processing to facilitate matching. Tokenization is different in that the query language is different from the document language. The tokens identified are intended for translation not for matching. English queries may need to be lowercased before dictionary lookup. Also, it is important to identify phrases before translation to minimize ambiguity.

2.5.3 Pre-translation query expansion

Since queries tend to be short, they often lack sufficient terminology to express the information need in all the different ways it might be expressed. A solution to this problem is to use query expansion and add relevant terms to the original query. The goal of CLIR pre-translation expansion goes beyond that of monolingual query expansion. Both attempt to find additional query terms to represent query concepts in multiple ways, however, in CLIR, pre-translation query expansion is also intended to increase the chance that all query concepts get translated (Levow, Oard, and Resnik, under review). The latter is very important because without a translation these concepts do not appear in the target query. Query expansion is typically done automatically using a document collection that shares the query language. Techniques mentioned are Local Context Analysis (Larkey, Ballesteros, and Connell, 2002; Martin and McCarley, 2002), and pseudo relevance feedback (Tomlinson, 2001), (Xu, Fraser, Weischedel, 2001), (Chen and Gey, 2001), (Darwish and Oard, 2002_2), (Chen and Gey, 2002), (Savoy and Rasolofo, 2002). These techniques are based on an initial retrieval run using the source query. The terms from the top *n* documents are ranked by their relevance weight and the top *n* terms are added to the original query. To expand English queries Larkey and Connell (2001) reported using AP news articles from 1994 through 1998. Larkey et al. (2002) used the Acrophyle system to expand acronyms. Another possibility is using a thesaurus for expansion. Manually created thesauri tend to add different terms to a query than collection-based relevance feedback.

It is important to point out that pre-translation query expansion has better results in languages that share alphabets (orthographies) because the expansions may include Named Entities that are potential cognates. The problem with pre-translation query expansion is that it can introduce erroneous query terms that are then potentially translated into multiple erroneous Arabic terms (Frasier, Xu, and Weischedel, 2002). Perhaps the greatest drawback to query expansion is that it results in longer queries which take longer to process (Tomlinson, 2002).

As mentioned previously, in addition to pre-translation query expansion, there is also the option to do post-translation query expansion (see section 5.5). Pre-translation expansion improves both recall and precision while post-translation expansion is intended to enhance precision.

2.5.4 Translation

As pointed out previously, translation plays an important role in CLIR. Since the query and the documents are in different languages, translation is needed to match across distinct vocabularies. Most of the TREC teams used a dictionary-based query translation approach where each query term is looked up in the translation resource and replaced by all or a subset of the available translations (Larkey, Ballesteros, and Connell, 2002; Gey and Oard, 2001). As a baseline approach, all the listed translations can be added, but it is advisable to carry out some form of disambiguation to cut down on query size. Another way to go about CLIR is to carry out document translation. In that case, translation takes place before, during or after indexing. However, most CLIR systems use query translation.

It is generally accepted that the translation causes a drop in performance although this is not always the case. The four main sources of translation knowledge that have been applied to CLIR are ontologies, bilingual dictionaries, machine translation lexicons, and corpora. In addition to these, translation can also be accomplished through transliteration (see section 5.4.6).

2.5.4.1 Missing translations

An important task in dictionary translation involves dealing with source terms that do not occur in the translation resource and are thus left without a translation. To increase the chances of finding a translation, Levow, Oard and Resnik (under review) propose using "backoff translation". Here a term that is not in the resource is divided into its smaller parts (in case of phrases) or stemmed to find morphological equivalents. As a last resort, terms can be transliterated (see section 5.4.6). Named Entities are often lacking in translation resources. Some CLIR Arabic track TREC participants created lists of frequently used Named Entities such as country and city names and had these manually translated or the transliterations checked (Larkey and Connell, 2001). NMSU created a dictionary with proper names that was used by some participants (Oard and Gey, 2002).

Post-translation query expansion (see section 5.5) is another way to recover some of the missing Named Entities.

2.5.4.2 Translation ambiguity

Another difficult task in dictionary translation is identifying the correct sense of the source word and then finding the correct translation for that sense of the source word. As a baseline in dictionary translation, one simply replaces the source term with all of its target translations, introducing many erroneous translations and creating a rather large target query. This effect is known as translation fan-out. Identifying phrases in the source query should cut down on the number of translation possibilities.

Many of the TREC CLIR Arabic track participants used translation probabilities (see section 5.4.3) to assign the likelihood of a translation being correct (i.e. (Xu, Fraser, Weischedel, 2001), (Darwish and Oard, 2002_2)). Participants were provided with a translation probabilities lexicon based on the U.N. parallel corpus and created by BBN. We have this dictionary among our resources. Researchers such as Pirkola have experimented with query formulation to include the different translation options rather than using translation probabilities. Pirkola (1999) used the InQuery System which has a synonym operator. He included all possible translations of a term in the synonym operator and thus used combined collection statistics for term counts, treating all terms as if they were one. Others used term frequency as a means of disambiguation, ranking the Arabic translations by frequency of occurrence in the Arabic collection, leaving out the less frequent terms (Chen and Gey, 2001). Darwish and Oard (2002_2) attempted what is called translation-based indexing where each instance of a term in a document is replaced with all of its translations at indexing time. To reduce the effect of common terms that have a lot of translations, they introduced balanced translation-based indexing where only the top 5 translations are indexed. A so-called two-phase translation was used by Aljlayl and Frieder (2001) where only the translations that correctly translated back into the source term are included as translations of that source term. They also experimented with adding only the first listed translation to the query as well as adding all translations listed. The two-phase translation technique outperformed the other two.

2.5.4.3 Translation probabilities

Many Arabic track CLIR TREC teams used translation probabilities to deal with translation ambiguity and term weighting issues, especially since a translation lexicon with probabilities was provided as a standard resource. However, most teams combined translation probabilities from different sources and achieved better retrieval results that way (Xu, Fraser, and Weischedel, 2002; Chowdhury et al. 2002). Darwish and Oard (2002_2) posit that since there is no such thing as a complete translation resource, you should always use a combination of resources and that your translation probabilities will be more accurate if you use more resources. Researchers that use Language Modeling (see section 6) use translation probabilities for the English language terms as well as for the Arabic (Frasier, Xu, and Weischedel, 2002).

2.5.4.4 Evidence combination

Research shows that combining translation resources increases CLIR performance (Larkey et al., 2002). Not only is your translation coverage increased, but your translation probabilities are more refined. Darwish and Oard (2002_2) used translation probabilities from different resources and combined all translation resources together by summing the different probabilities. Combining dictionaries is especially important when working with ambiguous languages.

2.5.4.5 Notes on translation resources

Dictionary coverage can also be increased by combining multiple translation resources together. Chen and Gey (2001) used a combination of dictionaries for query translation and compared retrieval performance of this dictionary combination with machine translation. The dictionaries outperformed MT. Small bilingual dictionaries were created by Larkey and Connell (2001) for place names who also inverted an Arabic-English dictionary to English-Arabic. They found that using dictionaries that have multiple senses, though not always correct, outperforms bilingual term lists with only one translation alternative.

When using and / or combining dictionaries it is important to note what base form the dictionary entries have. Some dictionaries use the singular form (for nouns) or indefinite form (for verbs), some use roots, others use stems. Free resources from the web often use a combination of all of the above. The resource entry form needs to be standardized across all resources and should match document processing.

A listing of the different translation resources can be found in sections 8.2, 8.3, and 8.8.

2.5.4.6 Transliteration

As a fall-back method for those terms that cannot be translated some researchers use pronunciation-based transliteration rather than leaving the term untranslated as is done in CLIR in languages that share orthographies. Darwish et al. (2001) mention that transliteration by MT systems is not very reliable and they provide the scheme they used to accomplish the task independently. English letters are mapped to the closest Arabic sounding letters For example, the letter “r” is mapped to ر (Fraser, Xu, and Weischedel, 2002; Darwish and Oard, 2002).

2.5.5 Post-translation expansion

Post-translation expansion is useful to reduce the effect of erroneous translations in the query (Aljlayl et al., 2001), and also to recapture some of the terms (i.e. Named Entities) that were lost in the translation. Post-translation expansion can be achieved in a similar fashion as pre-translation expansion (see section 5.3), but is carried out in the document language over the actual document collection.

2.5.6 Term weighting

Term weighting in CLIR is different from Monolingual Information Retrieval (MIR) due to the fact that query and documents do not share the same language. In MIR, query terms are weighted using the document and collection weights, something which can only be achieved indirectly in CLIR since the majority of the query terms are not present in the collection. One problem is that common query terms tend to have a large number of translations compared to less common terms. Because of this the common terms tend to get over-weighted. Another problem is that *idf* weighting (see section 4.6) tends to over-weight rare translations. Possible solutions to these two problems are explored by Darwish and Oard (2002_2) in the form of balanced translation and structured query translation.

2.6 Matching and evidence combination

The TREC Arabic CLIR track participants tended to use the same retrieval models as they used in monolingual information retrieval. The transformation of a monolingual system into a cross-lingual system typically focuses on the translation aspect, leaving the matching algorithms in place. For example, several researchers described using OKAPI's BM (Darwish and Oard, 2002_1; Savoy and Rasolofo, 2002) changing only, if at all, the constants in the algorithm. Exceptions to the status quo are the Language Modeling (LM) folks (Mayfield et al. 2001; Larkey and Connell, 2001). LM calculates the probability that the query is created through random sampling of the document. Documents are then ranked according to that probability. To extend the model to CLIR, the translation probability of an English source word, given the Arabic target translation, is included. After matching takes place, a ranked results list is generated.

Multiple TREC Arabic CLIR track teams merged results lists of different retrieval runs and found that this generally increased retrieval performance (Chen and Gey, 2001). This evidence combination approach, also known as data fusion, requires a merging strategy. These strategies range from simply summing the ranking scores of the individual runs (Savoy and Rasolofo, 2002) to complex score normalization functions (Larkey et al. 2002). Whether these performance improvements are actually noticeable to the user, and are thus worth the additional processing time, remains to be seen.

2.8 Evaluation

There have been two large scale Arabic retrieval evaluations as part of TREC. These Arabic tracks took place in 2001 (Gey and Oard, 2001), and 2002 (Oard and Gey, 2002) and had approximately 10 participating teams each.

2.8.1 TREC Arabic test collection

2.8.1.1 Documents

The Arabic tracks in TREC used the Linguistic Data Consortium Arabic news collection (LDC2001T55). This 400K collection contains 383,872 articles from Agence France Press (AFP) Arabic Newswire. The corpus is distributed as 2337 g-zipped files that result in 911,555,745 bytes (896MB) of text when uncompressed (Tomlinson, 2001). Each file consists of about 164 documents with an average size of 2375 (Tomlinson, 2001). The newspaper articles are encoded in UTF-8 and use non-vocalized Arabic which is more ambiguous than vocalized Arabic and can cause problems when trying to match with translation resources that are vocalized (Larkey, Ballesteros, and Connell, 2002). The documents originated between May 13 1994 and December 2000 (Chen and Gey, 2001). Chen and Gey (2002) found that after (minimal) normalization the collection contained 541,681 unique Arabic terms. This test collection was used in both TREC tracks.

2.8.1.2 Topics

Traditionally TREC topics consist of 3 parts: a title, description, and a narrative (see Figure 3). The title is the shortest version of a topic and describes it in a few key terms. The description is a longer version of a topic and describes the topic in a full sentence. The narrative lists explicitly what constitutes a relevant document and a non-relevant document. The narrative is intended to aid TREC judges in their relevance assessments but is often used as part of the topic for document retrieval.

For the Arabic track in TREC-2001, twenty-five topics were developed by LDC and NIST: AR1-AR25 (see Figure 3). The 2001 topics did not contain many proper names, and, unlike the *ad hoc* track, the title runs outperformed all other runs. The title fields were longer than the usual ad hoc track titles with 6 concentrated search terms. The 25 topics had on average 165 relevant documents per topic (Gey and Oard, 2001). For TREC-2002 an additional 50 topics were created: AR26-AR75 (not available to the public yet). The 50 topics had on average 118 relevant documents per topic. It is better to avoid using the AR1-AR25 topics for post-hoc evaluation because of issues with the relevance pool (see section 7.1.4). The AR26-AR75 topics are safe to use. However, query variability was very large so a query by query analysis is advisable (Oard and Gey, 2002).

The topics and the documents have different encoding schemes. The topics were encoded in ASMO 708 (or ISO 8859-6 (Tomlinson, 2002)) not UTF-8.

2.8.1.3 Relevance judgments

Relevance judgments were developed at LDC through relevance pooling (combining the top 70 documents from 20 runs submitted by 10 teams). The number of known relevant documents ranges from 6 to 556 per topic with an average of 165 relevant docs per topic. This is larger than the typical TREC collection and there is some indication that there may also be a substantial number of undiscovered relevant documents. The total number of relevant documents for the 25 topics of 2001 was 4,122. The average number of relevant documents per topic was 165 (100 is about the TREC norm) (Gey and Oard, 2001). Due to these circumstances there are some problems with the 2001 relevance data (see section 7.1.4 below).

Arabic Topic AR1

<top>
<num> Number: AR1
<title>
فنون العرض و المؤسسات الاسلامية في العالم العربي
<desc> Description:
ما هو اثر المؤسسات الاسلامية على فنون العرض مثل الرقص
و الموسيقى في العالم العربي؟
<narr> Narrative:
المقالات المتعلقة بالفنون الرياضية او التشكيلية
او بفنون العرض خارج العالم العربي
او بالسلوكيات الدينية خارج اطار فنون العرض
او بالديون و القروض المالية لا علاقة لها بالموضوع
</top>

English Topic AR1

<top>

<num> Number: AR1
<title> Performing Arts and Islamic Institutions in the Arab World

<desc> Description:
What is the impact of Islamic Institutions on the performing arts
such as dance and music in the Arab World?

<narr> Narrative:
Articles concerning sports or spatial arts, performance arts outside
of the Arab World, religious behavior not related to the performing arts,
or debts and banking loans, are not relevant to this topic.

</top>

Figure 3. AR-1 Arabic Topic and English Topic.

For the TREC2002 relevance pool, 41 runs were submitted and the top 100 documents of each run were added to the pool. The total number of relevant documents for the 50 topics of 2002 was 5,909. The average number of relevant documents per topic was 118. The number of relevant documents ranged from 10 to 523 (Oard and Gey, 2002).

2.8.1.4 Relevance pool issues

Relevance pooling is a technique to find most of the relevant documents in a collection without comparing all documents to a query. Rather, a pool of relevant documents is created based on the top *n* (usually 100) documents for each topic from each participant in the evaluation. The judges go through the relevance pools for each topic after the

removal of duplicates. The assumption is that most relevant documents are bound to be included in the pool and the rest of the collection does not need to be judged. All documents that are not included in the relevance pool are assumed to be not relevant. This process is called relevance pooling. For relevance pooling to work correctly one needs a large number and variety of search methods to produce the relevance pool and an adequate pool depth. The relevance pool created for the first Arabic track did not have many duplicate documents making for a very rich relevance pool and the assumption that there are many relevant documents left undetected. There are several reasons why this may be the case. The most obvious reason is the mismatch between the pool depth and the average number of relevant documents per topic. The pool depth was set at 70 and, as was discussed previously, the average number of relevant documents for these topics was 165. Another reason for the rich relevance pool is that the participants in this first time track tried many different techniques which resulted in unique sets of retrieved documents. Analysis by the track organizers revealed that the evaluation results are not stable and they caution against drawing conclusions from post-hoc experiments using this test collection.

2.8.2 Evaluation measure

The most commonly used retrieval effectiveness measure in IR evaluation is mean un-interpolated average precision (MAP) (Darwish and Oard, 2002). This measure supposedly reflects a user going down a list of retrieved documents in search of something relevant. The more relevant documents are found high in the list (relevant document density), the happier the user is going to be. MAP is not stable for queries with only a single known relevant document.

2.8.3 Retrieval performance

Retrieval performance in these evaluations varied. At least one team found that CLIR outperformed monolingual IR (Xu, Fraser, Weischedel, 2001) but this finding was not generally shared. Other teams reported a cross-lingual performance ranging from 66% (Sanderson and Alberair, 2001), 71-78% (McNamee, Piatko, and Mayfield, 2002), 85.6% (Chen and Gey, 2001), to 87.9% (Chen and Gey, 2002) when compared to their monolingual runs. Both Oard and Gey (2002) and McNamee, Piatko, and Mayfield (2002) report a relationship between CLIR retrieval performance and the quality and number of translation resources used. Generally speaking, the best results from 2002 are somewhat below those of 2001 (Oard and Gey, 2002). Perhaps this difference is due to the fact that the 2001 topics were easier because of the large number of relevant documents pre topic in the collection.

2.8.4 Other test collections

A test collection for OCR retrieval was created by Darwish and Oard (2002). The collection comes in a printed book called Zad Al-Me'ad. The researchers created several versions (representing varying levels of OCR accuracy) to build what is referred to as the Zad Collection (Darwish et al., 2001), provided by Al-Areeb Electronic Publishers, LLC.

It contains 4,000 documents and has 25 queries with relevance judgments. Queries are 3-6 words long in Arabic and English. Aljlayl and Frieder (2001) manually translated the English *ad hoc* topics of TREC-7 and TREC-9 into Arabic to query the English test collection. This way they could test Arabic-English retrieval performance.

3. The Arabic Information Retrieval (AIR) System

3.1 Introduction

Given the detailed technical description of the challenges of Arabic CLIR above, it is important to remember that in today's global, multilingual environment, information access capabilities are needed that can support users who have minimal understanding of the languages of the documents they need to search. This need provides the guiding principle in the Arabic Information Retrieval System (AIR) – to quickly translate an English query into the target language in order to facilitate initial matching and provide the user with helpful information as quickly as possible. To accomplish this, AIR focuses on high precision in the first few (e.g. 10) retrieved documents to provide the basis for pseudo-relevance feedback to select good Arabic terms to add to the query. High precision in the first, and succeeding rounds, comes from the use of rich and varied lexical resources to improve translation probabilities of initial query terms, and also to provide high-quality data for the interactive sense-disambiguation tool. The AIR System successfully combines several lexical resources into a single English-Arabic meta-resource for query translation and user-guided query disambiguation. The promising results of this automatic dictionary combination are now being extended into a general methodology that is highly portable to other languages and usable in a range of information access tasks.

The AIR system also includes research on transliteration and proper name (PN) detection. Since English and Arabic have different orthographies, PNs are typically transliterated and incorporated into the respective vocabularies. For example, we know Iraq's interim president as Ghazi Yawer, which is a transliteration of his Arabic name (غازي ياور). English PNs in the query that cannot be translated are automatically transliterated into Arabic using a probabilistic transliteration model. We also transliterate Arabic PNs from the documents into English for the document summary. Transliteration from Arabic to English is more complex as unvocalized Arabic text typically does not include the short vowels. While the resulting transliteration is likely phonetically correct, we need an additional validation step using the Levenshtein Distance algorithm (Levenshtein, 1966), to select a correct English equivalent.

Another unusual and valuable feature of the AIR design is the user's ability to group searches and search results into what we call Topics. The searches may have started with different English queries, and may have been conducted on different databases, but then relevance feedback for a new query can operate across the queries within a Topic. Topics are owned by individual users and persist between sessions. Topics and results can be shared between users, and queries can be scheduled by the owner to run regularly on new documents entering the system.

For future work, we plan on expanding the transliteration and PN recognition modules, adding entity extraction capabilities, and extending the system to additional languages.

3.2 Logging onto the AIR system

AIR is a sophisticated CLIR system providing English speakers with the ability to search Arabic content without knowledge of Arabic. AIR was designed with the experienced Intelligence Analyst (IA) in mind, while still maintaining ease of use. The AIR system goes beyond simple searching by providing the analyst with the ability to categorize and track Topics over time, automatically translate documents, archive professional translations, and improve searching by garnering user feedback at critical points in the search process.

System Features:

- Improved searching through relevance feedback
- Improved query accuracy through word sense disambiguation
- Extended search Topic management
- Integrated machine translation capabilities
- Proper Noun identification and transliteration
- Archival of document translations

The AIR User-Interface (UI) allows the user to communicate with the system. After an initial login screen (see Figure 4), the user enters the system.

Figure 4. AIR Login screen.

Unlike other search engines, the AIR system is designed to allow the user to track Topics of interest over time and organize the results into a comprehensive answer to the user's information need.

Using the AIR system revolves around understanding the relationship between *topics*, *enquiries*, and *searches*.

Simply put, a Topic is a broad description of the information need in general. It is used for organizing information over time and is not directly used in the search. An Enquiry is a specific query or question that is related to the Topic.

For example, as an analyst I may be interested in tracking the Muslim Brotherhood. More specifically I may want to know how it spread from Egypt to other countries, and what other names this organization is known as.

Topic:	Muslim Brotherhood
Enquiries:	1) The spread of the Muslim Brotherhood spread from Egypt to other countries
	2) Other names of the Muslim Brotherhood

A Search is an Enquiry that is executed against a specific database.

3.3 Topics

A Topic is a broad description of a user's information need in general. A Topic defines the general category under which the user is looking for information and serves as a placeholder for the specific Enquiries (queries) the user is going to ask the system concerning this general Topic.

3.3.1 Creating Topics

The *Create Topic* screen (see Figure 5) allows the user to enter a Topic title and a Topic description.

The Topic title should contain a few keywords that help the user to remember the general Topic by. In the description field the user can add more detailed information about the Topic.

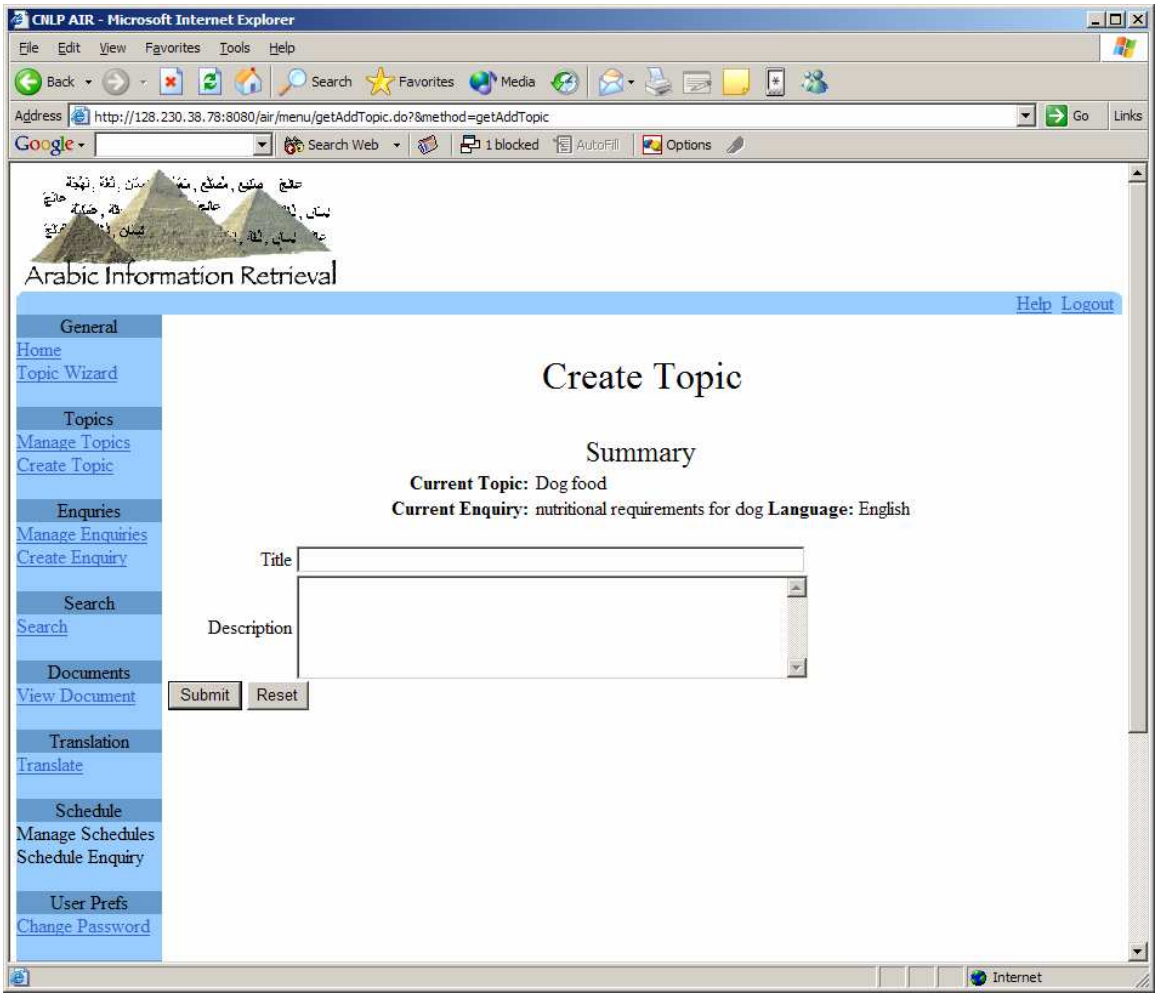

Figure 5. Create Topic screen.

3.3.2 Managing Topics

To manage existing Topics the user can click *Manage Topics* in the Topics section of the menu. The *Manage Topics* screen (see Figure 6) allows the user to perform several operations on their Topics:

1) select a Topic,
2) remove a Topic,
3) create a new Topic, and
4) edit an existing Topic

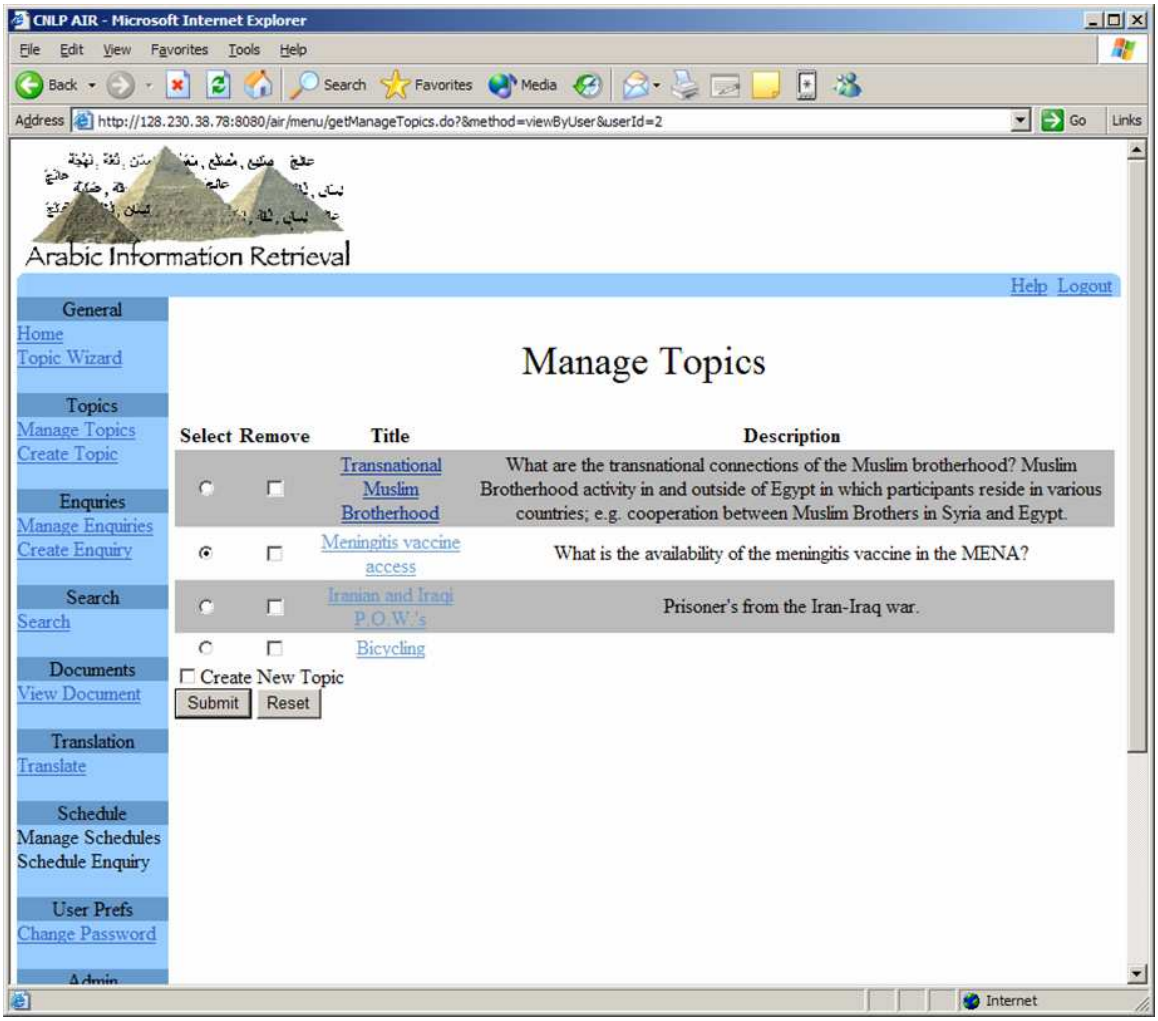

Figure 6. Manage Topics screen.

3.4 Enquiries

An Enquiry is a specific query or question that is related to the Topic.

3.4.1 Creating Enquiries

The *Create Enquiry* screen (see Figure 7) allows the user to enter an Enquiry for the Topic that he or she is currently working on. The screen displays their current Topic and, if applicable, the last Enquiry the user worked on within this Topic. The Enquiry is used to query the database.

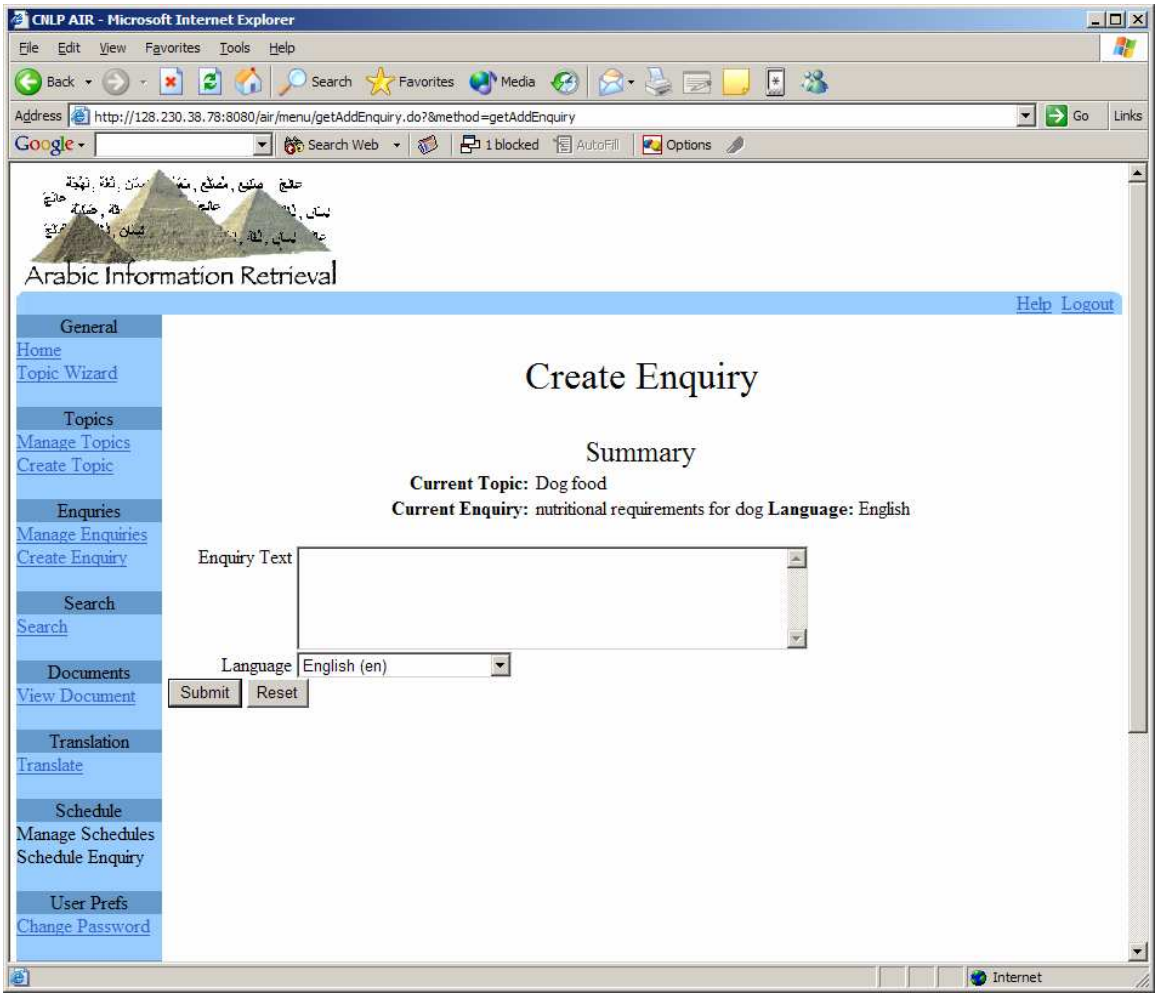

Figure 7. Create Enquiry screen.

3.4.2 Managing Enquiries

To manage existing Enquiries the user can click *Manage Enquiries* in the Enquiries section of the menu. The *Manage Enquiries* screen (see Figure 8) allows the user to perform several operations on his or her Enquiries:

1) select an Enquiry,
2) remove an Enquiry,
3) create a new Enquiry, and
4) edit an existing Enquiry.

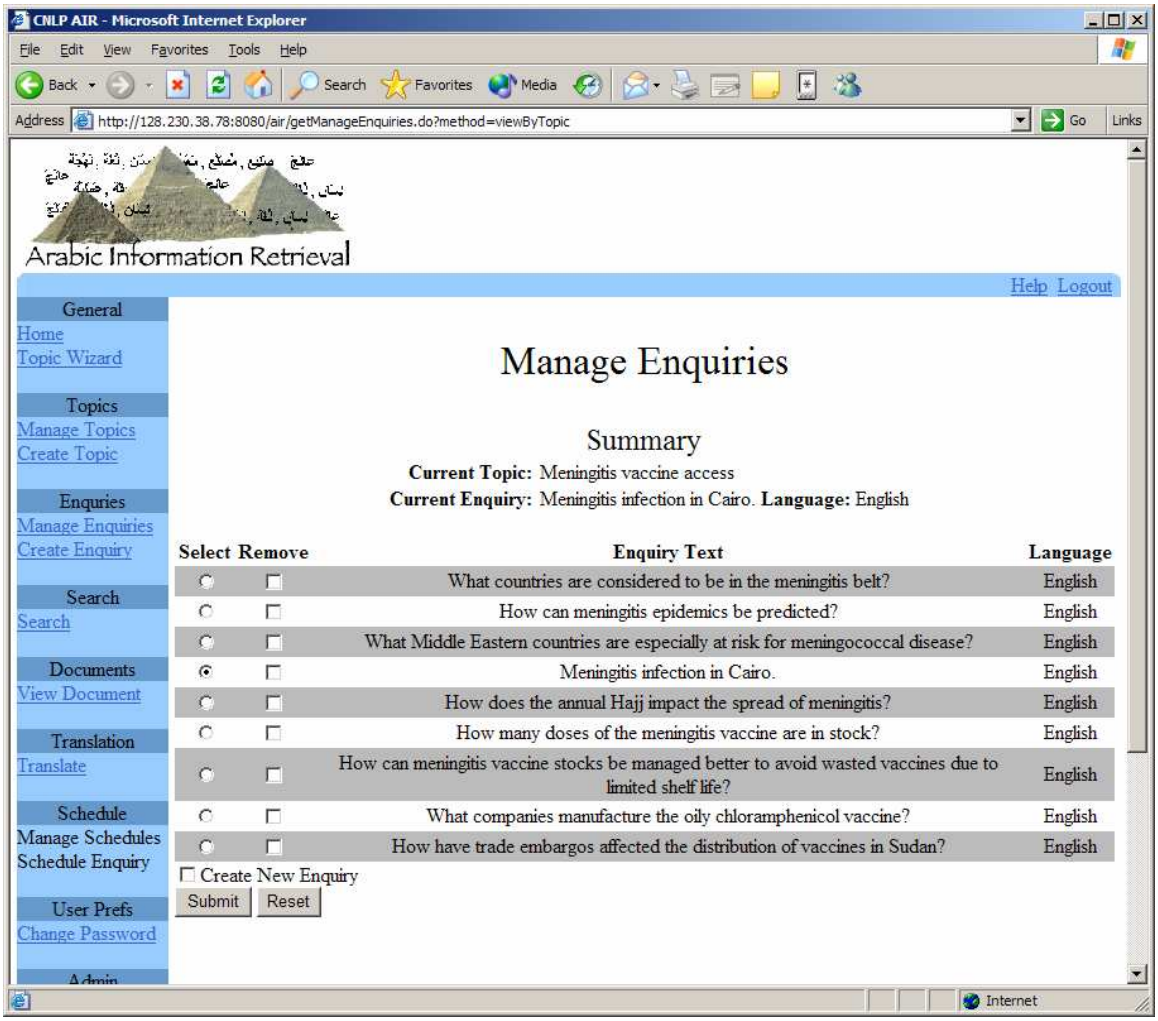

Figure 8. Manage Enquiries screen.

3.5 Performing a search

A Search is an Enquiry executed against a specific database with a given set of parameters for restricting the search (such as database, or date ranges, etc.).

To complete a search, the user first creates a Topic, then an Enquiry, and finally executes the Search, by selecting the database to be searched and disambiguating their query terms.

3.5.1 Query term disambiguation

After clicking the Submit button on the Search screen the user is brought to the *Disambiguation Translation* screen (see Figure 9). The *Disambiguate Translation* screen allows the user to confirm the correct sense of their query terms, rather than having the system second guess the intended meaning of these terms. Disambiguation of query terms is intended to improve the accuracy of their Enquiry.

For example, the search term “board” has multiple senses and when the system is selecting the correct translation for the term, it helps to know whether the user meant board as in “a wooden plank” or as in “a body of people”.

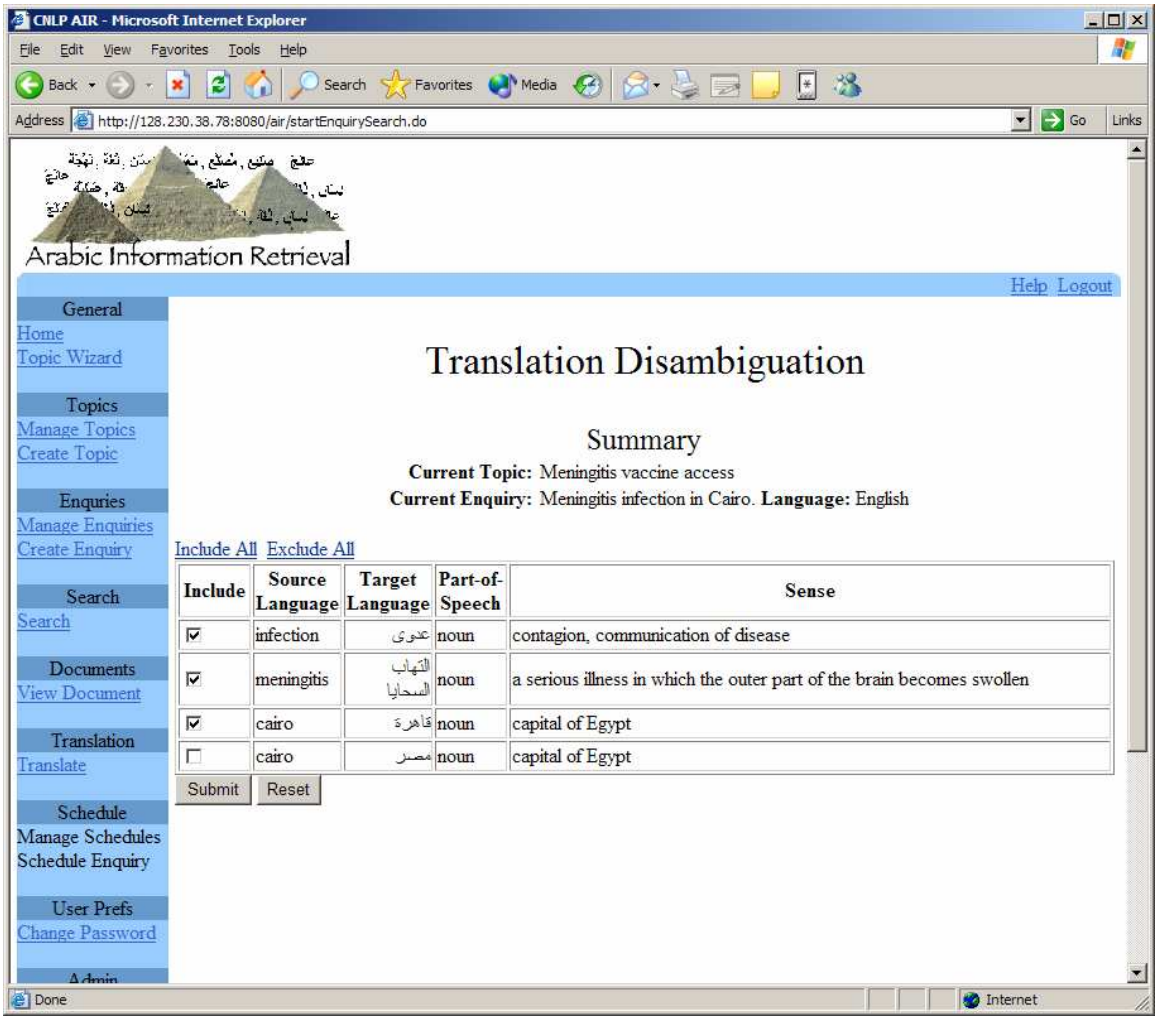

Figure 9. Translation Disambiguation screen.

3.5.2 Viewing Search Results

After selecting the right terms for an Enquiry the system will run the Enquiry against the database selected by the user. The results are displayed on the *Results* screen (see Figure 10).

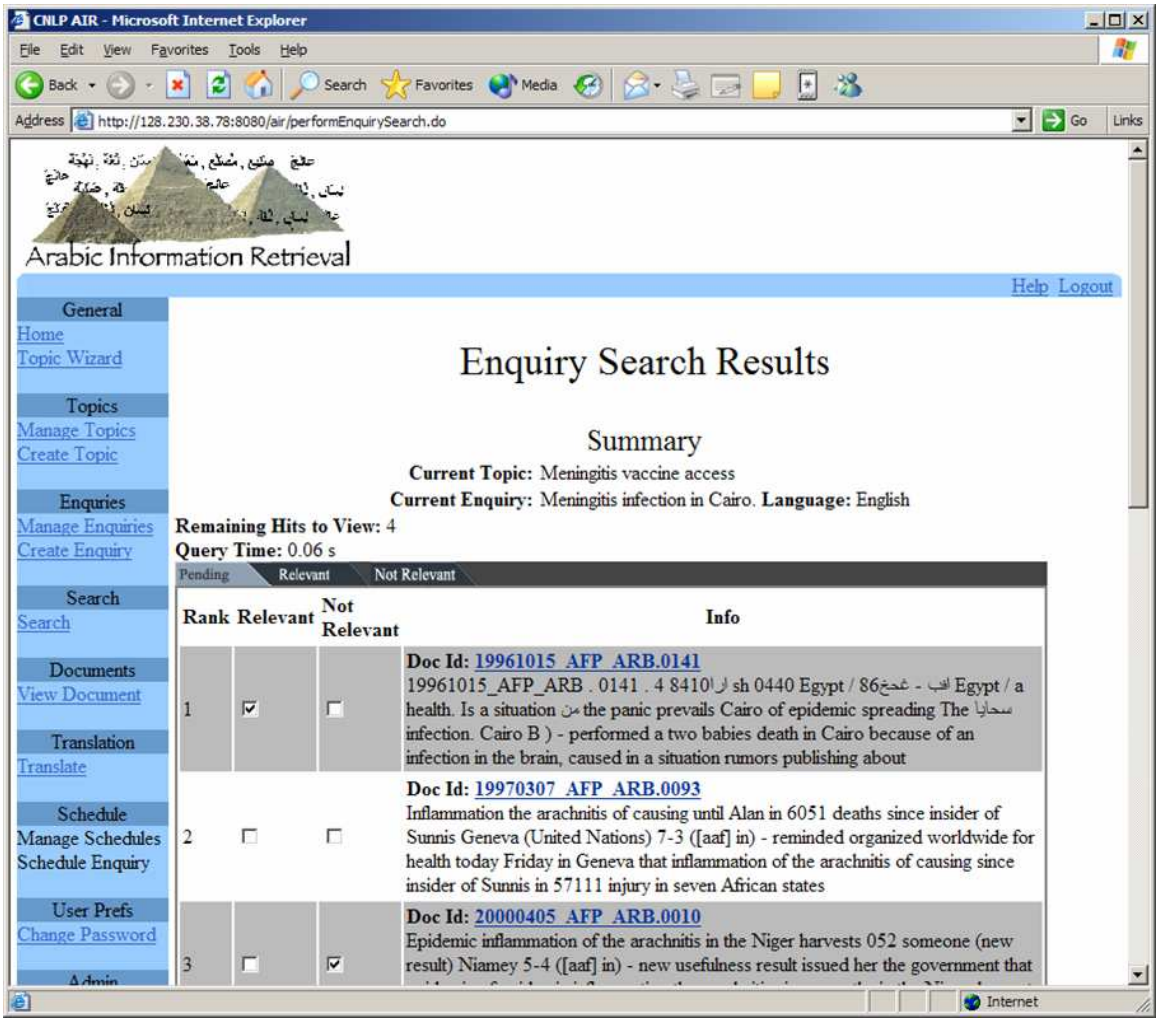

Figure 10. Results screen.

The retrieved documents are displayed in a list, the most relevant documents displayed first. By clicking on the document ID, the user can see the documents in an English translation as well as in their original language (see Figure 11).

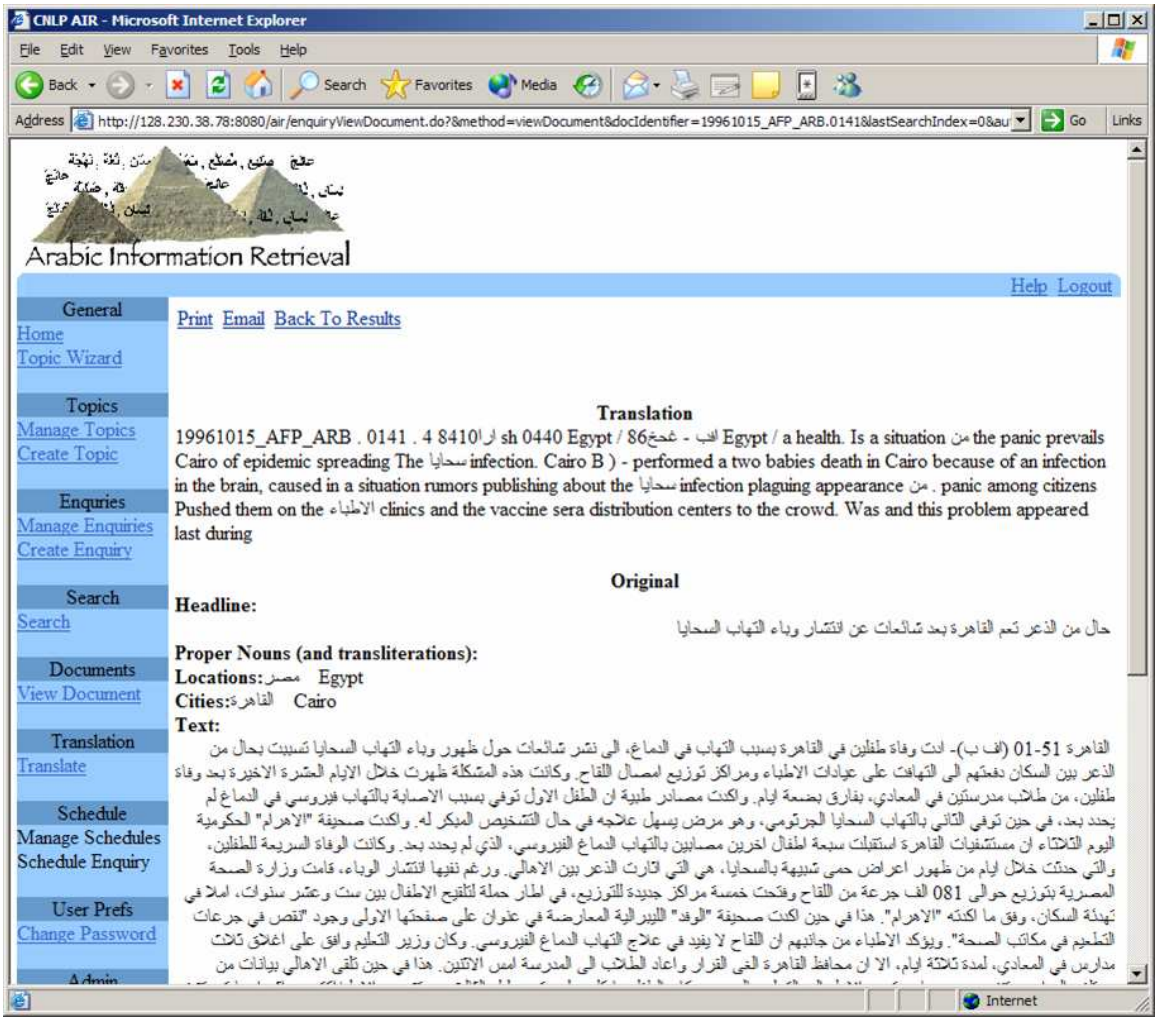

Figure 11. Document display screen.

3.5.3 User relevance feedback

When viewing the search results, the user can mark documents as relevant or non-relevant to their Enquiry. These relevance judgments are used to reformulate the user's original Enquiry. The improved Enquiry is then launched and the search results are updated starting with the next page of the current results.

3.5.4 Viewing previous results

Users can view their previous results for Enquiries as well as the documents they marked as relevant.

3.6 System administration

The system administrator can carry out the following user management related tasks:

1) View users or groups

Displays a list of all registered users or groups.

2) Add users or groups

Allows the administrator to add (register) new users or new groups.

3) Update users or groups

Allows the administrator to update information about users or groups.

4) Remove users or groups

Allows the administrator to remove users or groups from the system.

3.7 Other system features

3.7.1 Viewing individual documents

The AIR system can display documents directly. The user won't have to rerun a particular Enquiry to relocate a certain document. *View document* allows the user to type in the document ID and the system will display that particular document.

3.7.2 Translating Words

The AIR system allows users to utilize its internal dictionaries to search for a translation. The user types in the term(s) he or she wants to translate, selects how many translations they want to see, and selects which dictionary they want to use. The system displays the available translations.

3.8 System internals

3.8.1 Introduction

The AIR System is a three-tier web application built upon several popular, industry-validated, open source libraries along with our proprietary cross language search algorithm, and our unique Topic-based approach to IR based on intelligence analysts feedback. While the system is currently used to support searching in English and Arabic, the underpinnings of the system allow for searching in any language that has two key resources: content and MT capabilities. Furthermore, the MT capabilities can be as simple as a cross language dictionary or as sophisticated as a statistically trained machine translation system provided by a third party vendor. To date, we have demonstrated AIR using Arabic and Dutch and have additional MT capabilities for Korean and Chinese, but have not spent the resources to acquire content for these languages.

AIR is designed to be a flexible, scalable cross-language information retrieval system. To this end, many of the parameters in AIR are adjustable to fit an organization's needs. For instance, if memory is in short supply, AIR can load fewer dictionary items in order to conserve space. Furthermore, performance can be adjusted to processor speed, number of users, and index size by restricting the number of terms that go into a query, thus speeding up search.

3.8.2 Foundations

AIR is built on several best-of-breed technologies for providing a consistent, easy to use web interface along with fast search capabilities and a database independent persistence engine. These libraries, many of which are open source, are:

1. Java 1.4.2 (http://java.sun.com) -- Java provides a platform independent development language with a rich set of functionality.
2. Lucene 1.4.1 (http://jakarta.apache.org/lucene) -- A fast, flexible search engine based on the standard Vector Space Model. Lucene provides libraries for indexing, storing and searching large volumes of text.
3. Struts 1.1 (http://jakarta.apache.org/struts) -- The Struts framework provides mechanisms for quickly developing web applications based on the MVC (Model-View-Controller) development pattern. The MVC approach enforces a separation of work that allows for more flexibility when developing. Furthermore, it has a well-defined web template system for writing Java Server Pages (JSP).
4. Hibernate 2.2 (http://www.hibernate.org) -- Hibernate provides a database independent persistence engine and object serialization capability. It allows the programmer to work directly with Java objects without embedding raw SQL statements in the code. While we are currently using PostgreSQL (http://www.postgresql.org) for persistence, Hibernate allows us to easily switch to any of 14+ databases, including Oracle, Microsoft SQL Server and DB2.
5. Tomcat 5.0 (http://jakarta.apache.org/tomcat) -- Tomcat is used in the reference implementation for the Java Servlet and Java Server Pages technologies provided by Java. It is a reliable, well-documented server that easily integrates with many web servers.
6. Web Browser supporting HTML 4.0. AIR has been tested with both Internet Explorer 6.0 and Mozilla/Netscape.
7. A supported machine translation system. AIR supports: Systran 5.0 and Language Weaver. Other MT engines can easily be integrated.

All of these systems are utilized by AIR to provide a robust, easy to use cross language search capability that is designed to track and improve results over time.

3.8.3 System Requirements

AIR's system requirements are:

1. Java 1.4.x where x is greater than 2.
2. A Hibernate supported database. See http://www.hibernate.org/80.html for a list of supported databases. It is recommended that a separate user be set up in the database to hold the AIR schema.
3. The appropriate JDBC driver for the database of choice
4. 1 GB of memory, 2 GB optimal.
5. A Servlet/JSP container supporting Servlet Specification 2.3 or later, such as Tomcat 5.0.
6. Minimum Disk Space:
    a. 170 MB for the AIR application
    b. Space for database storage (depends on number of users supported, etc.)
    c. Space for Lucene indexes. Indexes are roughly 1.5 times the size of the original collection (including storage of the content).
7. Systran or Language Weaver MT System

3.8.4 System Architecture

As stated earlier, AIR is designed as a three-tier web application utilizing a web layer, an application layer, and a persistence layer.

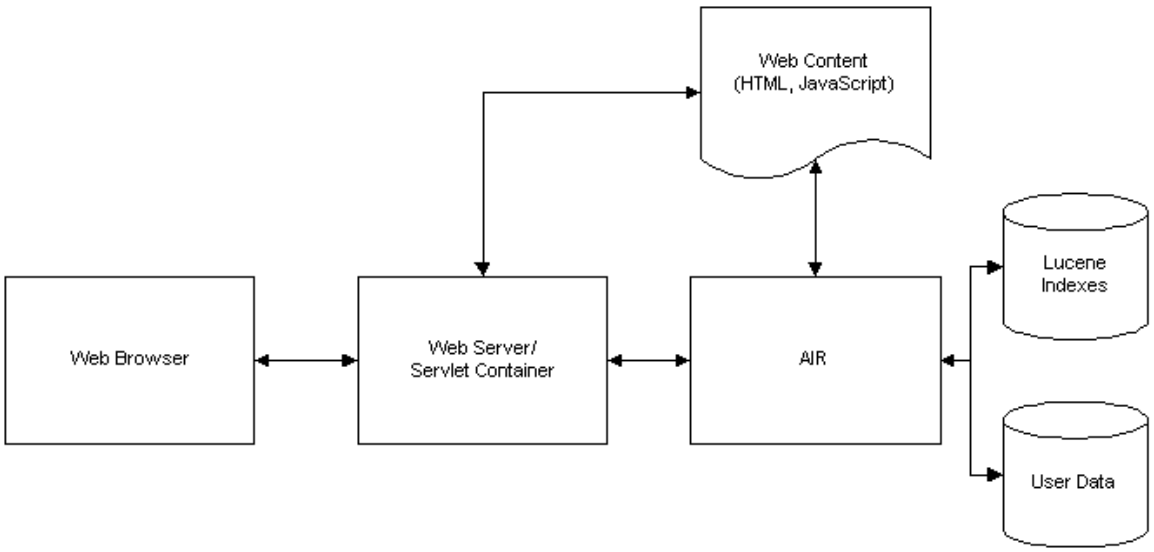

Figure 12. AIR System Architecture.

3.8.5 Search Process

Our search process can be pared down to the following pseudo code:

1. User inputs query in English
2. Analyze the English query, identifying any proper nouns in the query. For more on English Analysis, see the section on Indexing and Analysis below.
3. For each term or proper noun in the query:
    a. Get all translations sorted by likelihood
    b. If there are no translations, get all transliterations sorted by likelihood
4. Construct a Lucene-appropriate query by combining the highest weighted translations and transliterations for the proper nouns and terms
    a. Lucene provides many query operators to enable complex search possibilities. Of these, AIR utilizes the Boolean, phrase, boost and proximity operators. Currently, all tokens are OR-ed together, with each token optionally taking a boost factor depending on the value determined by AIR, as described above. Multi-token proper nouns are treated as phrases and have an optional proximity distance applied to them. (For instance, "Henry Thoreau"~3 would match "Henry David Thoreau" since Henry and Thoreau appear within 3 tokens of each other.)
5. Submit the query to Lucene and retrieve the initial results. Lucene analyzes the query in the same manner that the documents were analyzed so that terms are properly stemmed, etc. to match documents. (See the section on Indexing and Analysis below and section 5 for more on the Arabic analysis.)
6. Perform pseudo-relevance feedback by using the top X documents, where X is an adjustable parameter, to construct a new feedback query and resubmit to Lucene.
7. Send the results to be displayed in the UI.

8. As the user interacts with the results, they may mark documents as being relevant or non-relevant to their information need. As documents are marked, the system can improve on the initial results by doing a manual relevance feedback loop.
    a. Feedback loops are executed when the user has marked a minimum number of relevant documents. (currently 10 for the first time and 5 for all successive searches is executed.

The key steps in this process are the construction of the query with the appropriate weights and the manual feedback. Weights for the initial query are determined by looking up the frequencies of the translations and transliterations in the collection, and then setting the value to the term frequency divided by the maximum frequency of all of the translations. For example, if the word "bike" has three translations, with one occurring 100 times, the second occurring 50 times and the third occurring 25 times, the respective weights of the three translations will by 100/100 = 1, 50/100 = 0.5 and 25/100 = 0.25. Additionally, we can adjust the relative importance of proper nouns by weighting these terms separately.

During the manual feedback loop, the relevant and non-relevant documents are analyzed and used to reconstruct the query using a Rocchio model (Rocchio, 1971) adjusted to work with Lucene. This algorithm is described by the following pseudo code:

1. For each document in the relevant or non-relevant set:
    a. Get the top X terms from the document sorted by term frequency in the document and all terms in the query, where X is a parameter chosen during system setup.
    b. For each term
        i. If the system has seen the term previously, calculate the new weight using the formula $w_n = w_c + tf_i * rf$ where w_n is the new weight, w_c is the current weight, tf_i is the term frequency of the current term *i* in the current document and *rf* is the Rocchio Factor. The Rocchio Factor is α for terms in the original query β/|R| for terms in the relevant set and -γ/|NR| for terms in the non-relevant set where α, β and γ are adjustable parameters set through experimentation.
        ii. If the system has not seen the term before, calculate the weight of the term using the formula above with a w_c value of 0. Store the term in a lookup table.
    c. After all of the terms have been weighted, sort them by weight and use the top M terms, where M is an adjustable parameter, to construct a new query.
2. The new query is then submitted to Lucene, the system collects the results and filters out any results that were seen in previous searches, and presents them to the user as in the process outlined above.

3.8.6 Matching

Once queries have been formulated following the Search Process described above, the query is submitted to Lucene for processing. All matching is handled internally by

Lucene and documents are returned to AIR as a list of results, ordered by the score assigned by Lucene. Lucene employs an enhanced Vector Space Model (VSM) for ranking documents.

For the relevance feedback process, AIR utilizes Lucene's ability to store term vectors (the implementation of this was donated to Lucene by CNLP). Term vectors provide information about the terms in a specific document, such as term frequency and positions. This information is then used to construct the relevance feedback query described in the Search Process section.

For more information on how Lucene matches, refer to http://jakarta.apache.org/lucene.

3.8.7 Results Display

The display of search results in AIR is built upon a flexible system designed to work with multiple file structures while appearing transparent to the end user. AIR maps file extensions to different view handlers. Each view handler defines how to display that type of file in a web page. If a view handler is not defined, a default view handler is used that displays the Field text contents as defined in the Index. See the section on Indexing for more information on Fields. This approach to document views is used for viewing search results and the individual documents.

3.8.8 Indexing

Indexing is the process of taking a collection of documents, analyzing them using common Information Retrieval techniques, and then storing them in an inverted index maintained by Lucene. Lucene provides much of the indexing functionality, while this research provides the analysis modules for examining the text. (For more on Lucene, see http://jakarta.apache.org/lucene.) The basic indexing algorithm is summarized below:

1. For each Document in the collection
    a. For each subsection (called a Field by Lucene)
        i. Generate a stream of tokens (lexical units) by invoking the system provided analysis module. (See the Analysis section below for more information.)
        ii. For each token, Lucene calculates the appropriate statistics for a Vector Space Model and stores the information in its index.
2. Optimize the index to reduce the number of files used, etc.

3.8.9 Analysis

The Analysis module provided by AIR consists of a series of filters that are applied to the text. First the text is tokenized, which provides a stream of tokens that can then be marked and manipulated according to the system's IR needs. For tokenization, AIR uses a simple tokenizer that identifies a token as a consecutive series of Arabic letters (vocalized or non-vocalized), English letters, or digits (both Arabic and Hindu numerals). These tokens are then passed through the following filters:

1. Normalizer -- This filter strips off all vocalizations from the token.
2. Arabic Stop -- Removes Arabic stop words.

3. Hannouche Light Stemmer – A Light stemmer developed by CNLP. To learn more about the Hannouche stemmer, refer to section 5.

This process is also applied to any query that is executed against the Arabic database.

For query analysis in the cross language case, there are two applications of the analysis module, albeit using different tokenizers and filters. As stated above, the target language analysis must match the analysis used for indexing the document collection. AIR, however, also does analysis of the input language, in this case English, to improve on the translation results. For English, the following tokenizers and filters are used:

1. Standard Tokenizer – Provided by Lucene.
2. Possessive Stripper – Removes possessives from the token.
3. Proper Noun (PN) identifier – Identifies Proper Nouns in the query. PNs may consist of multiple tokens, in which case they are collapsed into a single token
4. Krovetz Stemmer[4] – A stemmer optimized for dictionary lookup. Stems are whole words and thus can be used for dictionary lookup.
5. English Stop – Removes English stop words. For English, we do this after PN identification to allow for PNs like *United States of America*.

The AIR system allows for easy changing of the tokenizers and filters used in analysis. It can also be easily extended to provide other types of analysis. The only caveat is that the analysis modules must be the same for indexing and searching.

4. System evaluation results

4.1 English Monolingual Retrieval Tests

Using TREC-7 English language document collection.

The major factors investigated were:

1. Stemming: The Porter stemmer is switched On or Off
2. Pseudo-relevance feedback (PsRF). After the initial search, the top T documents are assumed relevant for the purpose of the Rocchio feedback formula and the bottom B documents (of the top-ranked 100) are assumed non-relevant. PsRF is switched On or Off, and parameters are varied: Rocchio parameters α, β, and γ, T, B, and t/d = number of highest-weighted terms per document used. This factor represents a completely automatic step in the generation of the first ranked list of documents viewed by the user.
3. Simulated relevance feedback (SimRF). After the initial search and the PsRF search (if On), the top T documents are looked up in the TREC relevance file to determine whether they should be regarded as relevant or not, and the Rocchio formula is applied. SimRF is switched On or Off, and parameters are varied: Rocchio parameters α, β, and γ, T, and t/d = number of highest-

[4] R. Krovetz, 1993: "Viewing morphology as an inference process," in R. Korfhage et al., Proc. 16th ACM SIGIR Conference, Pittsburgh, June 27-July 1, 1993; pp. 191-202.

weighted terms per document used. This factor resembles the interactive judgment of documents by the user during a search session.

4.1.1 Evaluation measures

Precision is a measure that reflects the relevance of the retrieved documents. It is defined as the proportion of retrieved documents that are judged relevant by the user. In the TREC collection that we are using to test the system, there are 50 queries, and 472,676 documents. The average number of documents that have been specified in this test set as relevant to each query is 85.

Recall represents another aspect of retrieval effectiveness. At any given point in a search, recall is the proportion of relevant documents in the database that have been retrieved so far. As the search proceeds, recall increases monotonically from 0 to a maximum of 1.

The TREC evaluation procedure produces two tables of precision, averaged over all queries run:

1. Precision at each of 11 interpolated recall points (from 0 to 1.0). This is further summarized by the average of those 11 values (so called "Average Precision")
2. Precision when n documents have been retrieved (n = 5, 10, 15, 20, 30, 100, etc.)

At this stage, the research focus is the ability of the system to retrieve a good group of documents in the first few positions (because in Arabic, we will be depending on relevance feedback to improve the query.) In other words, we want high precision early in the search. So, Table 1 presents an evaluation measure to reflect this: the average of the first three precision figures in the second TREC table, namely precision values when 5, 10 and 15 documents have been retrieved. We'll call this "Early Precision".

Early in the search, recall will be very low, simply because few documents have yet been retrieved. Recall is of interest later in the search, so we report recall values when 30 and 100 documents have been retrieved, averaged over all queries.

4.1.2 Results

The results are listed in Table 1. Column Stems indicates whether the stemmer was switched on (●) or off. Sub-columns under PsRF give the values of parameters for pseudo-relevance feedback: if they are all blank, pseudo-relevance feedback was not used. Sub-columns under SimRF give the values of parameters for simulated relevance feedback: again, all blank means not used. The % EP difference column shows the EP differences between runs that vary in the setting of only one factor, *e.g.* in the case of run 2, the entry r1+1.5% means that early precision for run 2 was 1.5% greater than that for run 1. The difference between these two runs was the use of pseudo relevance feedback. The % Recall @ 100 column is similar, but based on recall when 100 documents have been retrieved. No significance tests have been performed on the differences. Interpretation of the results is based on the magnitude of the percentage differences. We

regard a difference of less than 5% as not noticeable in practice, and a difference of more than 10% as substantial.

4.1.3 Conclusions

In all cases where runs differ only in the application of the stemmer (r4 - r1, r5 - r2, r6 - r3, r12 - r11) substantial improvement (in both precision and recall) is gained by using the Hannouche stemmer. In other words, stemming is beneficial regardless of the use of pseudo-relevance feedback or simulated relevance feedback.

In cases where the only difference between runs is the use of simulated relevance feedback (r11 - r3, r12 - r6, r15 - r8, r16 - r9, r17 - r10), improvement in precision is gained by using simulated relevance feedback. In cases where the settings of pseudo-relevance feedback parameters are better (r15 - r8, r16 - r9, r17 - r10), the improvement is substantial, and recall is also improved. We conclude that simulated relevance feedback is beneficial regardless of the use of stemming and pseudo-relevance feedback.

In cases where the only difference between runs is the use of pseudo-relevance feedback (r2 - r1, r5 - r4, r7 - r4, r9 – r4, r16 – r33, r27 – r34), the difference in EP is only noticeable when the parameters are well-chosen. The use of pseudo-relevance feedback is favored. On the other hand, recall is improved by the use of pseudo-relevance feedback when it is used without simulated relevance feedback.

The positive effects of stemming and simulated relevance feedback on both Early Precision and Recall are cumulative. Pseudo-relevance feedback has little apparent benefit when simulated relevance feedback is used.

It is very likely that optimal parameter settings will vary from one situation to another. We report here findings for the English mono-lingual situation. The results (forthcoming) for Arabic mono-lingual and English-Arabic cross-language will probably be different.

Pseudo-relevance feedback parameters:

1. Smaller values of β (relative to α) are a little better than larger ones. Original query terms should be given greater weight than terms from the top few documents.
2. In conjunction with smaller β, smaller T and smaller t/d appear to be better.
3. Treating the lowest ranked documents as non-relevant ($B > 0$) diminishes performance.
4. The best values tried were $\alpha = 1$, $\beta = .15$, $T = 5$, $B = 0$, $t/d = 5$ (γ unused)

Simulated relevance feedback parameters:

1. Tests began with setting α, β, γ to the values (1, .5, .25) recommended by Rocchio in the early work on vector-based relevance feedback, and setting $T = 10$ and $t/d = 10$. The differences observed in both precision and recall are not large enough to be noticeable. Note also that in the operational system T will be determined separately for each query.

Run	Stems	PsRF						SimRF					Early Precision	Recall @ 30	Recall @ 100	% EP difference, w/ run	% Recall @ 100 difference, w/ run
		α	β	γ	T	B	t/d	α	β	γ	T	t/d					
1													.393	.099	.193		
2		1	1		10	0	10						.399	.103	.217	*(r1+1.5%)*	**r1+12.4%**
3		1	1		5	0	10						.430	.112	.223	***r2+8%***	(r2+2.8%)
4	●												.464	.111	.212	***r1+18%***	***r1+9.8%***
5	●	1	1		10	0	10						.486	.120	.246	*(r4+4.7%)*, **r2+21.8%**	**r4+16%, r2+13.4%**
6	●	1	1		5	0	10						.485	.120	.245	*(r5 - .06%)*, **r3+12.8%**	*(r5 - .4%)*, r3+9.9%
7	●	1	1		5	0	5						.470	.115	.235	*(r6 - 3.2%), (r4+1.29%)*	*(r6 – 4.1%)*, ***r4+10.8%***
8	●	1	.25		5	0	5						.491	.123	.247	*(r7+4.6%)*	r7+5.1%
9	●	1	.15		5	0	5						.499	.122	.248	*(r8+1.5%)*, r4+7.54%	*(r8+0.4%)*, **r4+17%**
10	●	1	.10		5	0	5						.505	.121	.247	*(r9+1.2%)*	*(r9 – 0.4%)*
11		1	1		5	0	10	1	.5	.25	10	10	.460	.112	.223	***r3+6.8%***	(= r3)
12	●	1	1		5	0	10	1	.5	.25	10	10	.519	.119	.246	*r6+6.88%,* ***r11+12.8%***	(r6+0.4%), ***r11+10.3%***
13	●	1	1		5	0	5	1	.5	.25	10	10	.529	.125	.251	***r11+15%***	***r11+12.6%***
14	●	1	.5		5	0	5	1	.5	.25	10	10	.550	.128	.260	*(r13+4%)*	*(r13+3.6%)*
15	●	1	.25		5	0	5	1	.5	.25	10	10	.574	.131	.266	**r8+16.8%**, *(r14+4.3%)*	r8+7.7%, *(r14+2.3%)*
16	●	1	.15		5	0	5	1	.5	.25	10	10	.584	.130	.269	**r9+17.1%**, *(r15+1.74%)*	r9+8.5%, *(r15+1.13%)*
17	●	1	.10		5	0	5	1	.5	.25	10	10	.572	.131	.271	**r10+13.4%,**	r10+9.7%,

																(r16 - 2%)	*(r16+0.7%)*
18	●	1	.15	.15	5	5	5	1	.5	.25	10	10	.531	.122	.248	r16 – 9.1%	r16 – 7.8%
19	●	1	.15	.1	5	5	5	1	.5	.25	10	10	.543	.122	.254	***r16 – 7%***	*r16 – 5.6%*
20	●	1	.15	.05	5	5	5	1	.5	.25	10	10	.552	.124	.249	r16 – 5.5%	r16 – 7.4%
21	●	1	.15	.1	5	10	5	1	.5	.25	10	10	.555	.127	.259	r16 – 5%**,** *(r19+2%)*	*(r16 – 3.7%)***,** *(r19+2%)*
22	●	1	.15		10	0	10	1	.5	.25	10	10	.560	.127	.267	*(r23 – 3.1%)*	*(r23 – 0.8%)*
23	●	1	.15		5	0	10	1	.5	.25	10	10	.578	.132	.265	*(r16 – 1%)*	*(r16 – 1.5%)*
24	●	1	.15		5	0	5	1	.4	.25	10	10	.584	.133	.271	*(=r16)*	*(r16+0.7%)*
25	●	1	.15		5	0	5	1	.6	.25	10	10	.589	.131	.267	*(r16+0.9%)*	*(r16 – 0.7%)*
26	●	1	.15		5	0	5	1	.7	.25	10	10	.587	.130	.267	*(r25 – 0.3%)*	*(= r25)*
27	●	1	.15		5	0	5	1	.6	.15	10	10	.586	.131	.266	*(r25 – 0.5%)*	*(r25 – 0.4%)*
28	●	1	.15		5	0	5	1	.6	.1	10	10	.588	.131	.265	*(r25 – 0.2%)*	*(r25 – 0.7%)*
29	●	1	.15		5	0	5	1	.6	.35	10	10	.586	.131	.266	*(r25 – 0.5%)*	*(r25 – 0.4%)*
30	●	1	.15		5	0	5	1	.6	.15	10	15	.573	.128	.262	*(r25 – 2.7%)*	*(r25 – 1.9%)*
31	●	1	.15		5	0	5	1	.6	.15	15	15	.581	.130	.256	*(r25 – 1.4%),* *(r30+1.4%)*	*(r25 – 4.1%),* *(r30 – 2.3%)*
32	●	1	.15		5	0	5	1	.6	.15	10	5	.562	.127	.260	*(r25 – 4.6%)*	*(r25 – 2.6%)*
33	●							1	.5	.25	10	10	.569	.127	.268	*(r16 – 2.6%),* **r11+23.7%**	*(r16 – 0.4%),* **r11+20.2%**
34	●							1	.6	.15	10	10	.583	.130	.271	*(r27 – 0.5%)*	*(r27+1.9%)*

Table 1. Results of English monolingual retrieval run.

4.2 Cross-language Evaluation

A series of "runs" of the English-Arabic system was performed in a non-interactive environment to assess the effects of the various retrieval system components. The document collection used in these tests is the TREC-10 collection of 383,872 news stories in Arabic together with 50 queries in English. In this database, 5909 documents are identified as relevant, *i.e.* an average of 118 documents per query (ranging from 3 to 523). Throughout the cross-language tests, the description form of the query was used. A "run" consists of searching the database for documents matching each of the 50 queries in turn. For each query, the documents are assigned scores and ranked. The evaluation is based on the top-ranked 100 documents for each query.

There are many adjustable parameters in the system which affect the document rankings, and the purpose of this evaluation was to determine the best settings of these parameters for the interactive system. There are aspects of an interactive environment which cannot be easily simulated in a non-interactive test platform, and these will be pointed out in due course. However, the non-interactive system affords us the opportunity to do a large number of runs much faster than could be done with a real user.

4.2.1 Definition of an evaluation Run

Each query is processed as follows:

1. The (English) query words are translated into Arabic using AIR's lexical combinatory resource. In this step, proper names (PNs) are identified, and words that are not found in the dictionary are transliterated instead.
2. *Optionally*, the Arabic query words are stemmed.
3. The initial query is constructed for processing by Lucene. Term weights are based on term frequencies in the query, individual documents, and the whole collection of documents (as described elsewhere).
4. A ranked list of documents is generated, and a cutoff applied at 100 documents.
5. *Optionally*, the top few (T) and the bottom few (B) documents are used to perform pseudo-relevance feedback, as described below. This results in a revised (Arabic) query which is used to re-rank the documents.
6. *Optionally*, the leading documents in the ranking are used to perform simulated user relevance feedback, as described below, resulting in a sequence of zero or more revisions to the query, each of which is used to re-rank the documents.

The final rankings for all 50 queries are processed by the TREC evaluation program to produce tables of precision and recall, from which we derive the performance summaries of interest for our purposes.

The runs are grouped into four sets:

Runs 1 – 42 are Initial search only (steps 1 – 4),
Runs 43 – 83 and 110 are Initial search and Pseudo-relevance feedback (steps 1 – 5),

Runs 84 – 94 are Initial search and Simulated relevance feedback (steps 1 – 4 and 6),
Runs 130 – 184 are Initial search, Pseudo-relevance feedback and Simulated relevance feedback (steps 1 – 6).

4.2.1.1 Pseudo-relevance feedback

The objective of this technique is to improve the query by picking up (Arabic) terms from the highest ranking documents retrieved by the initial search. Also, one may reduce the influence of terms occurring in low-ranking documents. The assumption is made that the top-ranked documents are relevant, and the bottom-ranked (of the 100 documents retrieved) are non-relevant. This is obviously not a very reliable assumption, however the technique has proven moderately effective in earlier mono-lingual experiments. The method used to generate the revised query is based on Rocchio's formula. In vector notation this is:

$$\underline{Q'} = \alpha\, \underline{Q} + \beta\, \underline{C_R} - \gamma\, \underline{C_N}$$

where $\underline{C_R}$ is the centroid of relevant documents retrieved (R) and $\underline{C_N}$ is the centroid of non-relevant documents retrieved (N). In Lucene, the effect of this formula has to be approximated by suitable setting of the "boost" factors in the query. In the course of this work, we implemented the formula in two ways, which are not mathematically equivalent. The original method scales the initial query, Q, by α first and then appends the weighted terms from R and N. The later method, which should be closer to the theory, constructs a merged list of terms from Q, R and N, and then calculates the weights. We call this the "integrated" method.

4.2.1.2 Simulated relevance feedback

This process simulates the interaction between a user and the retrieved documents. Instead of assuming that the top few documents are relevant, the program looks up the retrieved documents in the file of relevance judgments provided with the TREC collection (these were human judgments). The method of constructing a revised query is similar to that used for pseudo-relevance feedback, also based on the Rocchio formula. However, the document sets R and N are determined from the relevance file. The process is as follows:

1. The ranked list from the initial search is scanned, noting relevance judgments, until N-rel(1) relevant documents have been seen. Feedback is based on all the documents up to that point. The remaining documents in the database are re-ranked and appended to the documents already scanned. This is commonly called "residual ranking", and simulates what a user would actually see.
2. The new ranked documents are scanned until N-rel(2) new relevant documents have been seen. A new query is derived from the previous one and all documents retrieved up to this point. This is repeated until 100 documents have been "seen".

Note that in some cases, when the initial ranking contains fewer than N-rel(1) relevant documents among the top 100, relevance feedback will not be executed at all.

4.2.1.3 Parameters

Stemmer in step 2: the use of the stemmer is optional. In the Arabic monolingual tests we determined that the Hannouche stemmer is the best one available to us. The parameter is **Stem**, values **H** or none.

Translation in step 1 (two parameters):

Dict: the dictionary used, values **A** (Ajeeb), **C** (chained), **S** (sorted)

Ntrans: the maximum number of translations per word taken from the dictionary

Transliteration in step 1 (two parameters):

Ntranslit: the maximum number of transliterations per word

Translit non-PNs: Transliterate non-PN words that don't have translations, values **no**, **yes** (default)

Proper Names in step 1 (two parameters):

PN Boost: a number that (if present) replaces the weight assigned by the transliteration and lookup procedure

PN words: always include individual PN words (even when whole phrase is found): values **yes**, **no** (default)

Pseudo-relevance Feedback in step 5 (8 parameters):

Formula: the implementation of the Rocchio formula: values **O** (original), **I** (integrated)

T: the number of top-ranked documents assumed relevant

B: the number of bottom ranked documents (among 100 retrieved) assumed non-relevant

t/d: the number of (top-weighted) terms per document used for feedback

t/q: the number of (top-weighted) terms included in new query

α, β, γ: Rocchio formula parameters. (Note that **B** = 0 ↔ **γ** = 0)

Simulated Relevance Feedback in step 6 (8 parameters):

Formula: the implementation of the Rocchio formula: values **O** (original), **I** (integrated)

N-rel(1): number of top-ranked relevant documents used (first iteration)

N-rel(2): number of top-ranked relevant documents used (subsequent iterations)

t/d: the number of (top-weighted) terms per document used for feedback

t/q: the number of (top-weighted) terms included in new query

α, β, γ: Rocchio formula parameters.

4.2.2 Evaluation metrics

Precision is a metric that reflects the relevance of the retrieved documents. It is defined as the proportion of retrieved documents that are judged relevant by the user. Recall represents another aspect of retrieval effectiveness. At any given point in a search, recall is the proportion of relevant documents in the database that have been retrieved so far. As the search proceeds, recall increases monotonically from 0 to a maximum of 1.

The TREC evaluation procedure produces two tables of precision, averaged over all queries in a run:

3. Precision at each of 11 interpolated recall points (from 0 to 1.0). This is further summarized by the average of those 11 values (so called "Average Precision")
4. Precision when n documents have been retrieved (n = 5, 10, 15, 20, 30, 100, etc.)

The initial search will be followed by either a pseudo- or a user-based relevance feedback step. And, if it is used, the pseudo-relevance feedback step will be followed by user-based relevance feedback. Thus, for both feedback techniques to produce good revised Arabic queries, the system should retrieve some relevant documents in the high ranks in steps 4 and 5. In other words, for these steps, we want high precision early in the search. We developed a metric called "Early Precision (EP)" to reflect this requirement. EP is the average of the first three precision figures in the second TREC table, namely the precision values when 5, 10 and 15 documents have been retrieved.

Early in the search, recall will be very low, simply because few documents have yet been retrieved. Recall is of interest later in the search, so we report recall values when 30 and 100 documents have been retrieved, averaged over all queries. Average precision is a commonly used single measure of retrieval performance. Average precision increases as the relevant documents move closer to the top of the list, and as more relevant documents are included among (in our case) the top 100. After simulated relevance feedback, we do not expect much improvement in EP, because the residual ranking method does not affect the positions of documents retrieved early in the initial search. So, we depend on Average precision to evaluate simulated relevance feedback.

4.2.3 Results

Caveat: We are aware of the existence of software bugs that have caused inconsistent results to appear in some of the runs. So far, we have not been able to isolate and correct the bugs – they may be in the software we have written or in Lucene, the open source software we have used and adapted. The errors that we have detected so far have been very small, so we will report the results we have in the belief that finding and correcting the bugs will not produce any major changes.

1. Initial search

The purpose of the first set of runs (1 – 42) is to establish good values of the parameters for the initial search, *i.e.* those to do with stemming, translation, transliteration and proper names. Reasonable starting values were found while developing the software. The results are listed in Table 2.

- The clearest result from this set of runs is that Early precision is substantially better when stemming is applied to the Arabic words (H in the Stem column). This appears

to hold, whatever the values of other parameters: EP for stemming is better than EP for no stemming by 31% - 45%.

- The Ajeeb (A) dictionary performed slightly better than chained (C) and sorted (S) dictionaries. However, this result is probably not a good predictor of performance in the interactive situation because the chained and sorted dictionaries were designed in order that a display of word senses might be generated for the user to select the most promising translations. This feature cannot be simulated in the test environment.

- Middle values (10, 20) of Ntrans appear slightly better than high or low values.

- Low values (2, 3) of Ntranslit seem marginally better than high.

- Run 41 indicates that non-Proper names that have no translations in the dictionary should be transliterated.

- Applying a PN Boost does not improve performance.

- Always including individual PN words, even when whole phrase is found (PN words) makes very little difference (Run 42)

- The settings of Run 38 were retained for most of the runs testing the various forms of relevance feedback (Tables 3, 4, 5)

2. Pseudo-relevance feedback

The second set of runs (Table 3) explore the parameters involved in the Pseudo-relevance feedback technique.

- The first result is that with the better parameter settings, pseudo-relevance feedback improves the best EP obtained in the initial search by 10% - 12%.

- The original implementation (O) of the Rocchio formula is somewhat better than the integrated version (I), although this is not consistent across all variations in other parameters.

- T, the number of top-ranked documents regarded as relevant was best set at 7. No results are available for B, the number of low ranked documents considered non-relevant, in the cross-language situation.

- t/d = 10 and t/q = 30 appear marginally better than other values.

- Based on results in the mono-language tests, B was set to zero for these tests, implying that $\gamma = 0$. The ratio of α to β is then all that needs adjusting. Thus in these runs, β is always 1 and α varies from 4 down to 0. $\alpha = 3$ gives slightly better results

than other values. Strangely, $\alpha = 0$ (*i.e.* disregarding the initial query) works quite well.

- Using a different set of parameters for the initial search (R36, which also performed quite well), produced lower values of EP after pseudo-relevance feedback (Runs 82, 83).

3. Simulated relevance feedback without pseudo-relevance feedback

In the third set of runs (Table 4), simulated relevance feedback is applied immediately after the initial search, with no intervening pseudo-relevance feedback step.

- Average precision is substantially improved over that obtained in the better initial searches (by 18% – 38%).

- The original implementation (O) of the Rocchio formula seems to perform better than the integrated one (I). The other results are rather difficult to interpret.

4. Pseudo-relevance feedback and Simulated relevance feedback

The fourth set of runs (130 – 184) explored the parameter settings for simulated relevance feedback, following two of the better pseudo-relevance feedback runs (R58 and R110) – see Table 5.

The integrated implementation (I) of the Rocchio formula was used for most of these simulated relevance feedback runs, and is the one used in R58, because it is more faithful to the formula.

- Performance of the original implementation (O) in terms of average precision is substantially better (runs 178 - 184). In these O runs there is also a noticeable improvement over the runs in set 2, using only pseudo-relevance feedback. The better I runs perform only a little better than R58 in set 2.

- Better results are found with N-rel(1) = 4 and N-rel(2) = 3. These are the numbers of relevant documents required before feedback is performed, in the first and subsequent iterations, respectively. If these numbers are larger, feedback is delayed too long, or may not occur at all. On the other hand, if they are smaller, there are too few relevant documents on which to base the revised query.

- The performance differences obtained between various values of t/d and t/q are small and not very consistent. 10 seems to be a slightly better value than 20 for t/d.

- The ratios α:β and β:γ are adjusted in these runs. β remains fixed at 1. The values $\alpha = 2$, $\gamma = 0.75$ appear to be better. As in the pseudo-relevance feedback case, $\alpha = 0$ works quite well for simulated relevance feedback. The explanation is presumably that the initial Arabic query is frequently of poor quality, due to translation problems.

4.2.4 Parameter settings recommended for interactive system

Stemmer:

Use Hannouche

Translation:

Dict: Sorted. (Ajeeb performed better, but cannot support word-sense display, the use of which can be expected to improve performance)

Ntrans: the maximum number of translations per word = 20

Transliteration:

Ntranslit: the maximum number of transliterations per word = 2

Translit non-PNs: Transliterate non-PN words that don't have translations = yes

Proper Names:

PN Boost: do not use

PN words: always include individual PN words = no

Pseudo-relevance Feedback:

Formula: the implementation of the Rocchio formula = **O** (original)

T: the number of top-ranked documents assumed relevant = 7

B: the number of bottom ranked documents (among 100 retrieved) assumed non-relevant = 0

t/d: the number of (top-weighted) terms per document used for feedback = 10

t/q: the number of (top-weighted) terms included in new query = 30

α, β, γ: Rocchio formula parameters = (3, 1, 0)

Simulated Relevance Feedback:

Formula: the implementation of the Rocchio formula = **O** (original)

N-rel(1): number of top-ranked relevant documents used (first iteration) = 4

N-rel(2): number of top-ranked relevant documents used (subsequent iterations) = 3

t/d: the number of (top-weighted) terms per document used for feedback = 10

t/q: the number of (top-weighted) terms included in new query = 30

α, β, γ: Rocchio formula parameters = (2, 1, 0.75)

Run	Stem	Dict	Ntrans	Ntranslit	PN Boost	Other	EP	R at 30	R at 100	Ave.P	EP Comparisons
1		C	100	3			.141	.031	.076	0.0486	
2	H	C	100	3			.188	.037	.085	0.0498	33% > R1
3		A	100	3			.141	.031	.076	0.0487	
4	H	A	100	3			.188	.037	.085	0.0513	33% > R3
5		S	100	3			.142	.030	.069	0.0432	
6	H	S	100	3			.185	.036	.082	0.0423	30% > R5
7		C	10	3			.131	.031	.074	0.0479	
8	H	C	10	3			.190	.038	.085	0.0488	45% > R7
9		A	10	3			.133	.031	.075	0.0481	
10	H	A	10	3			.190	.038	.085	0.0503	43% > R9
11		S	10	3			.140	.029	.073	0.0428	
12	H	S	10	3			.192	.036	.078	0.0405	37% > R11, 6.7% > R24
13		C	5	3			.132	.030	.074	0.0479	
14	H	C	5	3			.183	.038	.085	0.0465	39% > R13
15		A	5	3			.133	.030	.075	0.0480	
16	H	A	5	3			.183	.038	.085	0.0468	38% > R15
17		S	5	3			.129	.026	.069	0.0360	
18	H	S	5	3			.174	.033	.077	0.0369	35% > R17
19		C	10	10			.137	.032	.076	0.0476	
20	H	C	10	10			.188	.037	.087	0.0470	37% > R19
21		A	10	10			.137	.032	.076	0.0477	
22	H	A	10	10			.188	.037	.087	0.0472	37% > R21
23		S	10	10			.137	.028	.066	0.0389	
24	H	S	10	10			.180	.034	.077	0.0369	31% > R23
25	H	S	10	3	1.5		.184	.038	.085	0.0433	(4.2% < R12), 10% > R26
26	H	S	10	3	5		.167	.035	.083	0.0384	
27	H	C	20	3			.187	.038	.086	0.0503	
28	H	A	20	3			.193	.038	.086	0.0518	

29	H	S	20	3			.185	.036	.079	0.0411	
30	H	C	15	3			.188	.038	.084	0.0486	
31	H	A	15	3			.188	.038	.084	0.0500	
32	H	S	15	3			.184	.036	.079	0.0401	
33	H	C	30	3			.173	.035	.079	0.0401	
34	H	A	30	3			.184	.038	.086	0.0519	
35	H	S	30	3			.181	.036	.078	0.0414	
36	H	S	10	2			.200	.036	.081	0.0446	9.9% > R37
37	H	S	10	1			.182	.032	.077	0.0356	
38	H	A	20	2			.207	.039	.090	0.0552	7.3% > R28, 12.5% > R39
39	H	A	20	1			.184	.035	.083	0.0406	
40	H	A	20	2	1.5		.199	.040	.095	0.0520	(3.9% < R39)
41	H	A	20	2		Translit non-PNs = no	.164	.032	.078	0.0436	21% < R38
42	H	A	20	2		PN words = yes	.204	.038	.088	0.0531	

Table 2. Cross-language Initial Search Runs.

Run	Initial Search	Formula	T	B	t/d	t/q	α, β, γ	EP	R at 30	R at 100	Ave.P	EP Comparisons
43	R38	I	5	0	10	30	1, 1, 0	.219	.045	.098	0.0558	
44	R38	O	5	0	10	30	1, 1, 0	.217	.048	.112	0.0609	
45	R38	I	5	5	20	30	2, 1, 1	.200	.044	.098	0.0561	
46	R38	O	5	5	20	30	2, 1, 1	.218	.049	.114	0.0667	
47	R38	I	5	0	10	30	2, 1, 0	.211	.044	.097	0.0555	
48	R38	I	5	0	10	30	.5, 1, 0	.199	.041	.091	0.0534	
49	R38	I	5	0	10	30	.25, 1, 0	.199	.043	.093	0.0527	7.6% > R68
50	R38	I	5	0	10	30	0, 1, 0	.228	.046	.100	0.0540	14% > R69
51	R38	I	5	0	10	30	3, 1, 0	.211	.044	.097	0.0574	
52	R38	O	5	0	10	30	2, 1, 0	.216	.046	.108	0.0630	
53	R38	O	5	0	10	30	.5, 1, 0	.217	.048	.106	0.0597	9% > R48
54	R38	O	5	0	10	30	.25, 1, 0	.212	.045	.096	0.0555	6.5% > R49
55	R38	O	5	0	10	30	0, 1, 0	.212	.044	.095	0.0535	7% < R50
56	R38	O	5	0	10	30	3, 1, 0	.227	.048	.114	0.0647	7.6% > R51
57	R38	O	5	0	10	30	4, 1, 0	.213	.045	.098	0.0558	
58	R38	I	7	0	10	30	3, 1, 0	.229	.048	.100	0.0616	8.5% > R51, 8% > R59, 10.6% > R38
59	R38	I	7	0	10	30	2, 1, 0	.212	.045	.094	0.0602	
60	R38	I	7	0	10	30	1, 1, 0	.222	.046	.099	0.0623	
61	R38	I	7	0	10	30	.5, 1, 0	.226	.046	.099	0.0626	13.5% > R48
62	R38	I	7	0	10	30	.25, 1, 0	.221	.046	.098	0.0611	11% > R49
63	R38	I	7	0	10	30	0, 1, 0	.227	.046	.095	0.0590	
64	R38	I	3	0	10	30	3, 1, 0	.205	.043	.096	0.0531	
65	R38	I	3	0	10	30	2, 1, 0	.204	.043	.095	0.0526	
66	R38	I	3	0	10	30	1, 1, 0	.205	.043	.094	0.0518	
67	R38	I	3	0	10	30	.5, 1, 0	.205	.044	.097	0.0521	
68	R38	I	3	0	10	30	.25, 1, 0	.185	.039	.090	0.0473	10.6% < R38
69	R38	I	3	0	10	30	0, 1, 0	.200	.043	.093	0.0497	

70	R38	I	5	0	20	30	0, 1, 0	.206	.043	.090	0.0527	7.6% < R50
71	R38	I	5	0	20	50	0, 1, 0	.206	.044	.091	0.0527	
72	R38	I	5	0	30	30	0, 1, 0	.196	.043	.088	0.0502	
73	R38	I	5	0	30	50	0, 1, 0	.196	.043	.088	0.0502	
74	R38	I	7	0	20	30	3, 1, 0	.218	.046	.098	0.0602	
75	R38	I	7	0	20	50	3, 1, 0	.212	.045	.095	0.0581	
76	R38	I	7	0	30	30	3, 1, 0	.217	.045	.096	0.0598	
77	R38	I	7	0	30	50	3, 1, 0	.215	.045	.094	0.0602	
78	R38	I	5	0	8	30	0, 1, 0	.200	.041	.092	0.0505	12% < R50
79	R38	I	7	0	8	30	3, 1, 0	.217	.044	.096	0.0614	5.2% < R58
80	R38	I	5	0	10	20	0, 1, 0	.206	.043	.092	0.0518	9.6% < R50
81	R38	I	7	0	10	20	3, 1, 0	.226	.047	.099	0.0613	9.7% > R80
82	R36	I	5	0	10	30	0, 1, 0	.200	.042	.095	0.0575	12% < R50
83	R36	I	7	0	10	30	3, 1, 0	.188	.039	.085	0.0489	18% < R58
110	R38	O	7	0	10	30	3, 1, 0	.231	.049	.115	0.0690	(0.9% > R58), 12% > R38

Table 3. Cross-language Pseudo-relevance Feedback Runs.

Run	Initial	Formula	N-rel(1)	N-rel(2)	t/d	t/q	α, β, γ	EP	R at 30	R at 100	Ave.P	Ave.P Comparisons
84	R38	O	4	3	10	30	2, 1, 0.5	.242	.053	.130	0.0763	18% > R85, 38% > R38
85	R38	I	4	3	10	30	2, 1, 0.5	.220	.048	.103	0.0648	
86	R38	O	4	3	20	50	1, 1, 0.5	.213	.045	.108	0.0649	
87	R38	I	4	3	20	50	1, 1, 0.5	.220	.046	.103	0.0623	
88	R38	I	4	3	20	30	3, 1, 0.5	.221	.046	.099	0.0630	
89	R38	I	4	3	20	30	2, 1, 0.5	.228	.048	.104	0.0651	3.3% > R88, 18% > R38
90	R38	I	4	3	20	30	1, 1, 0.5	.227	.046	.101	0.0634	
91	R38	I	4	3	20	30	0.5, 1, 0.5	.233	.045	.092	0.0573	9.6% < R90
92	R38	I	4	3	20	30	0.25, 1, 0.5	.234	.046	.106	0.0633	10.5% > R91
93	R38	I	4	3	20	30	0, 1, 0.5	.240	.044	.101	0.0605	
94	R38	O	4	3	20	30	2, 1, 0.5	.234	.053	.127	0.0747	15% > R89

Table 4. Cross-language Simulated Relevance Feedback Runs (without Pseudo-relevance Feedback).

Run	Initial	Formula	N-rel(1)	N-rel(2)	t/d	t/q	α, β, γ	EP	R at 30	R at 100	Ave.P	Ave.P Comparisons
130	R58	I	4	3	10	30	3, 1, 0.5	.237	.048	.097	0.0627	
131	R58	I	4	3	10	30	2, 1, 0.5	.242	.048	.103	0.0641	
132	R58	I	4	3	10	30	1, 1, 0.5	.229	.049	.109	0.0638	13.5% > R133
133	R58	I	4	3	10	30	0.5, 1, 0.5	.231	.045	.093	0.0562	13.5% > R134
134	R58	I	4	3	10	30	0.25, 1, 0.5	.236	.045	.084	0.0495	
135	R58	I	4	3	10	30	0, 1, 0.5	.230	.044	.105	0.0603	22% > R134
136	R58	I	4	3	10	50	3, 1, 0.5	.238	.048	.097	0.0629	
137	R58	I	4	3	10	50	2, 1, 0.5	.245	.048	.101	0.0639	
138	R58	I	4	3	10	50	1, 1, 0.5	.229	.049	.109	0.0662	20% > R139
139	R58	I	4	3	10	50	0.5, 1, 0.5	.230	.046	.093	0.0552	20% > R140
140	R58	I	4	3	10	50	0.25, 1, 0.5	.225	.045	.083	0.0461	
141	R58	I	4	3	10	50	0, 1, 0.5	.249	.045	.105	0.0588	28% > R140
142	R58	I	4	3	20	30	3, 1, 0.5	.250	.047	.094	0.0611	
143	R58	I	4	3	20	30	2, 1, 0.5	.257	.048	.099	0.0624	
144	R58	I	4	3	20	30	1, 1, 0.5	.251	.050	.105	0.0638	12% > R145
145	R58	I	4	3	20	30	0.5, 1, 0.5	.247	.046	.093	0.0568	31% > R146
146	R58	I	4	3	20	30	0.25, 1, 0.5	.236	.042	.076	0.0433	
147	R58	I	4	3	20	30	0, 1, 0.5	.252	.044	.102	0.0589	36% > R146
148	R58	I	4	3	20	50	3, 1, 0.5	.252	.047	.093	0.0610	
149	R58	I	4	3	20	50	2, 1, 0.5	.244	.045	.093	0.0607	
150	R58	I	4	3	20	50	1, 1, 0.5	.240	.047	.100	0.0611	7.6% > R151
151	R58	I	4	3	20	50	0.5, 1, 0.5	.246	.047	.092	0.0568	18% > R152
152	R58	I	4	3	20	50	0.25, 1, 0.5	.244	.045	.081	0.0480	
153	R58	I	4	3	20	50	0, 1, 0.5	.234	.044	.103	0.0577	20% > R152
154	R58	I	3	3	20	30	3, 1, 0.5	.229	.047	.091	0.0586	
155	R58	I	3	3	20	30	2, 1, 0.5	.258	.048	.099	0.0628	7.2% > R154, 5.4% > R173
156	R58	I	3	3	20	30	1, 1, 0.5	.237	.047	.097	0.0613	18% > R157
157	R58	I	3	3	20	30	0.5, 1, 0.5	.223	.042	.079	0.0520	12% > R158

158	R58	I	3	3	20	30	0.25, 1, 0.5	.218	.042	.078	0.0465	
159	R58	I	3	3	20	30	0, 1, 0.5	.233	.043	.097	0.0576	24% > R158
160	R58	I	4	2	20	30	3, 1, 0.5	.238	.046	.082	0.0562	
161	R58	I	4	2	20	30	2, 1, 0.5	.248	.047	.094	0.0603	7.3% > R160
162	R58	I	4	2	20	30	1, 1, 0.5	.251	.048	.100	0.0621	8.8% > R163
163	R58	I	4	2	20	30	0.5, 1, 0.5	.246	.047	.091	0.0571	21% > R164
164	R58	I	4	2	20	30	0.25, 1, 0.5	.234	.042	.084	0.0472	
165	R58	I	4	2	20	30	0, 1, 0.5	.247	.045	.101	0.0611	29% > R164
166	R58	I	4	3	20	30	3, 1, 0.75	.244	.047	.093	0.0609	
167	R58	I	4	3	20	30	2, 1, 0.75	.251	.047	.099	0.0617	
168	R58	I	4	3	20	30	1, 1, 0.75	.247	.044	.106	0.0637	16% > R169
169	R58	I	4	3	20	30	0.5, 1, 0.75	.240	.043	.092	0.0547	31% > R170
170	R58	I	4	3	20	30	0.25, 1, 0.75	.224	.040	.071	0.0417	
171	R58	I	4	3	20	30	0, 1, 0.25	.240	.042	.089	0.0537	29% > R170
172	R58	I	4	3	20	30	3, 1, 0.25	.238	.045	.090	0.0592	
173	R58	I	4	3	20	30	2, 1, 0.25	.240	.042	.089	0.0596	
174	R58	I	4	3	20	30	1, 1, 0.25	.241	.048	.094	0.0599	6.8% > R175
175	R58	I	4	3	20	30	0.5, 1, 0.25	.247	.046	.090	0.0561	22% > R176
176	R58	I	4	3	20	30	0.25, 1, 0.25	.239	.041	.079	0.0460	
177	R58	I	4	3	20	30	0, 1, 0.25	.248	.044	.092	0.0547	19% > R176
178	R110	O	4	3	10	30	2, 1, 0.5	.247	.052	.124	0.0756	9.6% > R110, 18% > R131
179	R110	O	4	3	10	50	1, 1, 0.5	.234	.050	.124	0.0722	4.6% > R110, 9.1% > R138
180	R110	O	4	3	10	30	2, 1, 0.75	.249	.053	.125	0.0771	11.7% > R110
181	R110	O	4	3	10	30	2, 1, 1	.245	.052	.124	0.0759	10% > R110
182	R110	O	3	3	10	30	2, 1, 0.75	.245	.052	.123	0.0740	
183	R110	O	2	3	10	30	2, 1, 0.75	.255	.054	.125	0.0746	
184	R110	O	5	3	10	30	2, 1, 0.75	.236	.051	.125	0.0731	

Table 5. Cross-language Simulated Relevance Feedback Runs (with Pseudo-relevance Feedback).

4.3 User Evaluation

A user study was carried out with three analysts. The goal of the study was to get user feedback from real users who had not been involved in the AIR project. Each session of the user study was preceded by a brief introduction to the system. The users watched while the researcher showed and explained different aspects of the system. After this introduction, the user was asked to read a brief introductory statement about the system and to complete a number of tasks on the system (see Appendix 1). Each user was given their own personal login ID to make the sessions as realistic as possible and to give the analysts the sense of having a space on the server for their Topics and Enquiries as actual users would. During the task assignments users were asked to think-aloud and share any thoughts, ideas, and problems they had in relation to the AIR system. After each session there was time for a debriefing.

The user study informed improvements to the system and provided suggestions for future work as well. In general, the analysts all liked the system. They found the user interface easy to use and the pages easy to read. This was borne out in the study as all users were capable of using the system after only a brief introduction. Users also liked the way the system kept track of prior searches and the way it organized them within a Topic. The issues that came up during the evaluation are listed below. Most have been fixed, others require work that is beyond the scope of this project but could be completed when we acquire future funding.

	User comments	**status**
Translation		
	Keeping track of manual translations once uploaded.	future
	Words that cannot be translated by Machine Translation are transliterated which makes it hard to look them up in a dictionary.	fixed
	Only part of the document is being translated.	fixed
	Can the user add words to the dictionary?	future
Enquiry formulation		
	Why not put a search button here?	fixed
	Could we get a spellchecker for system input?	future
Translation disambiguation		
	“Include all”, “exclude all” buttons don’t work.	fixed
	Color code translations that are synonyms to speed up selection process.	future
Search		
	Emailing results doesn’t work.	fixed
	Printing results doesn’t work.	fixed
	Would be nice to know more about search algorithm.	future
	Sort prior results by relevance judgments rather than original rank.	fixed
	Submit button doesn’t do anything after you’ve created your new Enquiry.	fixed
General UI comments		
	The word translation in the menu is confusing.	fixed

	Would be nice if the keywords could be highlighted.	future
	Years are displayed in reverse.	fixed
	Keywords may not be in search results as only the first paragraph of the document is displayed.	future
	Browse Documents option is not really browsing.	fixed
	Starting screen doesn't tell you where to start.	fixed
	Non wizard search process is disjointed. Hard to traverse the screens.	fixed
Help files		
	Help files don't mention how to create an Enquiry in an existing Topic.	fixed
	Upload translations not explained in help files.	fixed
Relevance Feedback		
	Check marks for relevant documents disappear.	fixed
	Unclear if relevance judgments are saved when pressing certain buttons on this screen.	fixed
	Documents in second search had relevance judgments from prior search.	fixed
	Risky to use only the first paragraph to judge relevance on.	future
	If there is no translation available - the user should not be allowed to give relevance feedback.	future
	Give instructions to not give feedback if unsure about relevance.	future
Search history		
	Results are displayed in Arabic - should be in English.	fixed
	It should be made clearer that you can sort these results in different ways.	fixed
	Tabs do not work on history screen.	fixed

5. Discussion

In this section we describe aspects of research and development that went into the AIR system such as dictionary combination, stemming, transliteration and proper name detection.

5.1 Dictionary combination

Cross-Language Information Retrieval (CLIR) systems facilitate matching between queries and documents that do not necessarily share the same language. To accomplish this matching between distinct vocabularies, a translation step is required. The preferred method is to translate the query language into the document language by using machine translation, or lexicon lookup. While machine translation may work reasonably well on full sentences, queries tend to be short lists of keywords, and are often more suited for lexical lookup (Oard and Diekema, 1998).

Part of our research for the AIR project involved the creation of a lexical translation resource to aid in query translation. At the same time this lexical resource provides the user of the system with lexical semantic information about each of the possible translations to

aid with disambiguation of the Arabic query (see section 3.5.1 query disambiguation). We created this lexical resource through the combination of an English-Arabic dictionary with WordNet. For future work we plan on folding in additional dictionaries, to increase dictionary coverage and mapping accuracy.

Research shows that combining translation resources increases CLIR performance (Larkey et al., 2002) Not only does this combination increase translation coverage, it also refines translation probability calculations. Chen and Gey used a combination of dictionaries for query translation and compared retrieval performance of this dictionary combination with machine translation (Chen and Gey, 2001). The dictionaries outperformed MT. Small bilingual dictionaries were created by Larkey and Connell (2001) for place names who also inverted an Arabic-English dictionary to English-Arabic. They found that using dictionaries that have multiple senses, though not always correct, outperform bilingual term lists with only one translation alternative. Combining dictionaries is especially important when working with ambiguous languages such as Arabic.

Many TREC teams used translation probabilities to deal with translation ambiguity and term weighting issues, especially since a translation lexicon with probabilities was provided as a standard resource. However, most teams combined translation probabilities from different sources and achieved better retrieval results that way (Xu, Fraser, and Weischedel, 2002), (Chowdhury et al., 2002), (Darwish and Oard, 2002_1). Darwish and Oard (2002_1) posit that since there is no such thing as a complete translation resource one should always use a combination of resources and that translation probabilities will be more accurate if one uses more resources.

The current CNLP dictionary, created by mapping together an English-Arabic dictionary and WordNet, used a combination of evidence to ensure the correctness of these mappings. In short, to create dictionary mappings, our lexicon combination module opens the dictionary files to be combined and cycles through the base dictionary entry by entry. For each entry it checks the other dictionary for possible mappings. Once a group of matching candidates has been established, mapping feature vectors are created for each.

Mapping feature vectors combine evidence of the mapping into a single vector which will later be used for mapping score calculation. The mapping with the highest score is the one that is going to be chosen for the mapping, unless this score is lower than a certain threshold. In case an entry cannot be matched, it is used as is, to increase dictionary coverage. The mapping feature vectors have 12 elements, which can be organized into three groups: ambiguity indicators, string similarity indicators, and sense likelihood indicators.

The mapping feature vector ambiguity indicators measure the ambiguity within the target dictionary, within the part-of-speech of the term, and of the mapping. The first vector element counts the number of possible matches in the other dictionary. A larger number of mappings indicates the presence of ambiguity, whereas a low number of mappings indicates a more obvious match between entries. The second vector element includes the number of votes for a certain entry. As the entry in the base dictionary contains a number of English translations, each translation can be considered to have a vote for the possible mapping. The

more English translations from the base dictionary point to the same dictionary entry in the target dictionary, the more likely the two entries should be matched. The third vector element indicates how many entries match in the target dictionary for the same part of speech as the base dictionary. The more matches, the more ambiguity and the more difficult it is to achieve a reliable match. The fourth vector element indicates how many translations were used as evidence for the mapping. The more evidence the more likely the outcome is correct.

The string similarity indicators measure how alike certain parts of the dictionary entries are. The more similar the entries, the more likely it is they should be mapped. The similarity is calculated using one of three different matching coefficients: the cosine similarity coefficient, the dice coefficient, and a straight matching coefficient. Each of these coefficients has unique qualities and a selection is made on the nature of the items we are trying to match. There are five vector elements that indicate string similarity. They compare the WordNet lexical file name, the WordNet headword (if available), the WordNet gloss, the WordNet definition, and the WordNet synonyms with the English definition from the base dictionary.

The sense likelihood indictors measure the commonality of the dictionary entries that are being mapped. Some word senses are more common than others and if both dictionary entry senses are common, they are more likely to be a match. Vector elements indicate the WordNet sense within a part-of-speech, the base dictionary sense within a part-of-speech, and the WordNet concordance count.

In the following example the program tried to map the Arabic terms for genus into WordNet:

> genus: group of animals or plants within a family, جِنْس,عِرْق,عُنْصُر,جنس,عرق,عنصر
>
> There are two possible mappings:
>
> 1) genus; noun; 2; group; 00; 1; (biology) taxonomic group containing one or more species
> 2) genus; noun; 1; cognition; 00; 2; a general kind of something ;"ignore the genus communism"
>
> The program created the following mapping feature vectors with final mapping scores (last element):
> 1) 2 1 2 1 na 0.18 0 2 1 1 1 0 536
> 2) 2 1 2 0 na 0.00 0.00 1 1 2 1 0 20
>
> The mapping with the highest score was selected resulting in the mapping:
> genus; noun; 2; جنس,عرق,عنصر;(biology) taxonomic group containing one or more species; 536

The mapping feature vectors (created for all possible mappings) are used to calculate the final matching score for each possible mapping. The score is calculated using a weighted combination of the feature vector elements. The mapping with the highest score is chosen as the final mapping. The dictionary combination program has been evaluated informally to select the best settings for matching score calculations. For future work we plan to fold in an additional dictionary and carry out a full scale evaluation.

5.2 Transliteration

The AIR project also included research and development in the area of transliteration. Transliteration is the representation of Arabic characters in letters of the English alphabet and vice versa. Since English and Arabic have different orthographies, proper names (PNs) are typically transliterated and incorporated into the respective vocabularies. For example, we know Iraq's interim president as Ghazi Yawer, which is a transliteration of his Arabic name (غازي ياور). English PNs in the query that cannot be translated are automatically transliterated into Arabic using a probabilistic transliteration model. We also transliterate Arabic PNs from the documents into English for the document summary. Transliteration from Arabic to English is more complex as unvocalized Arabic text typically does not include the short vowels. While the resulting transliteration is likely phonetically correct, we need an additional validation step using the Levenshtein Distance algorithm (Levenshtein, 1966), to select a correct English equivalent.

The AIR transliteration research is divided into two areas that are similar but not identical in their approach: English to Arabic transliteration, and Arabic to English transliteration. In both cases we perform an initial lookup to check whether the source term occurs in the database that holds translations and transliterations for frequently occurring proper names such as country names. In those cases where the term is listed, no further transliterations are needed. If the term is not listed, it is decomposed into separate parts that are defined in the respective transliteration models. Each part of the original term is replaced by the most likely corresponding equivalent in the target language. This likelihood is determined probabilistically.

Our English to Arabic model builds on research by AbdulJaleel and Larkey (2003), while our Arabic to English model has been developed by CNLP. The English to Arabic model was expanded to include character combinations that were previously disregarded (i.e. "Chr" now maps to "كر") in the previous model, this mapping did not exist.

Since mappings are not unique, the result of transliterating one single name leads to multiple possible names in the target language. Additionally, since the spelling in Arabic texts is not consistent, many variations of the same name may exist. To deal with this problem we validate each transliteration using the text and select only the first two validated transliterations with the highest score. An example of this procedure can be found below. The transliterations are sorted according to their weight; shown in the first column. Here we are only showing the top 10 generated transliterations. The values 1 and 0, after the weight value, indicate the validity of this translation.

#Christian
5.1 1 كريستيان
4.7000003 0 كريستايان
4.7000003 1 كريستان
4.7 0 كرايستيان
4.7 0 كرستيان
4.7 0 كريزتيان
4.7 0 كريصتيان
4.7 0 كريسطيان
4.7 1 كريستين
4.6 1 كريستيين

We developed a new Arabic to English transliteration model based on statistical analysis of the morphology of English proper names. The model maps each Arabic character, or group of characters, to their corresponding English character and assigns a weight to this mapping. A co-occurrence analysis was carried out on a test set of names to determine which characters should be grouped together. Since many Arabic characters can be mapped to more than one English character, the weighting is based on the frequency of this mapping in the test set of English names. The results of this analysis are then normalized and the weights are assigned accordingly. It is important to note that this morphological analysis only included English proper names, leaving out, for example, Latin names such as Octavio Paz. For future work we'll consider a larger selection of name types to improve the transliteration. Adding more name types will significantly increase the complexity of a transliteration model.

Since Arabic does not have capital letters and letters at the end of a word may have a different mapping than the same letters in the middle of a word, the model consists of three parts: initial letters, medial letters, and final letters. Each part of the model contains a mapping for a single character, or set of characters based on their position in the word. Table 6 illustrates the different mappings of the same Arabic letter depending on its position in the word for the first letter of the Arabic alphabet, the Alif (ا).

Position		
Start	Middle	Last
ا -> A 0.50	ا -> a 0.78	ا -> a 1.0
ا -> E 0.30	ا -> o 0.11	
ا -> I 0.1	ا -> e 0.09	
ا -> O 0.05	ا -> au 0.02	
ا -> U 0.05		

Table 6. Transliteration weights for the Alif.

Modern standard Arabic texts tend to be unvocalized which means that the short vowels, normally added as diacritical marks, are left out from the text. This poses problems for the Arabic to English transliteration model as characters that are not present in the word cannot be transliterated. Our transliteration model accommodates this problem and creates a so called "fuzzy" transliteration, which reflects the pronunciation of the English proper name but not necessarily its correct spelling. To capture the correct spelling we have added a validation step using an English newspaper corpus of roughly the same time period as its Arabic counterpart. The validation step uses the Levenshtein Distance algorithm (Levenshtein, 1966) to compare the fuzzy transliteration to actual English proper names to obtain the correct spelling of that name.

The following example illustrates how the procedure works before and after the validation step. The name to be transliterated from Arabic into English is روبيرتو افنتاخادو (Roberto Aventajado).

The results without the validation step are: *Robirto Avntakhado*, *Robirto Afntakhado, Robirto Aphntakhado, Robirto Evntakhado, Robirto Avntakhadu, Robirto Affntakhado, Robirto Efntakhado, Robirto Avntakhadw, Robirto Awntakhado.* Note that these results, though phonetically correct, require a validation step to select the correct spelling of the proper name. During the validation step a comparison with an English newspaper takes place and the system concluded that the correct spelling of this transliteration is: Roberto Aventajado.

5.3 Stemming

In our project, we developed a new Arabic stemmer which is a modification of the Light8 Stemmer (Larkey, Ballesteros, and Connell, 2002). A study by Larkey, Ballesteros, and Connell (2002) showed that the Light8 stemmer has better performance than any other light stemmer in cross-language retrieval. The Light8 stemmer was shown to outperform other stemmers when using unexpanded as well as expanded queries, as Light8 had a higher early precision. In the AIR system, using our own stemmer, named the Hannouche Stemmer after our native Lebanese researcher, Jean Hannouche, gave even better average precision than the Light8 stemmer.

In the Hannouche stemmer, a modification of the Light8 Stemmer, the normalization step is now part of the stemmer and has the same rules as Light8. However, the list of suffixes and prefixes were changed to accommodate different and more cases than the Light8. Light8 has three rules that need to be applied. We use this set of rules but in a different order and with small changes. This was done to make these rules even more effective.

To test our Stemmer, we ran 3 different experiments. Each experiment used one stemming technique and 3 different query types: Title (T), Description (D), and Narrative (N). Title queries are very short, often a few keywords only, Description queries are about a sentence in length, whereas Narrative queries may be up to a paragraph in length.

The experimental results are illustrated in the following table.

Experiment	Query Text	Stemmer	Early Precision	Recall @ 30	Recall @ 100
1	T	No	0.259	0.058	0.141
	D	No	0.237	0.052	0.128
	N	No	0.186	0.042	0.103
2	T	Light 8	0.309	0.07	0.162
	D	Light 8	0.325	0.066	0.156
	N	Light 8	0.229	0.047	0.114
3	T	Hannouche	0.313	0.07	0.163
	D	Hannouche	0.334	0.066	0.156
	N	Hannouche	0.236	0.049	0.122

Table 7. Different Stemming Results.

We first ran the different types of queries using no stemmer at all. Then we repeated the same experiments using the Light8 stemmer and the Hannouche stemmer. The results show that the Hannouche Stemmer outperformed the Light8, although by a small percentage. It is important to note that the execution time of both stemmers is about equal so there are no extra costs connected to using the Hannouche stemmer versus the Light8 stemmer.
Under the same conditions and parameters, the retrieval early precision was higher for the Hannouche Stemmer than for Light8. Both stemmers had the same recall value at 30 documents and 100 documents.

5.4 Proper Name detection

The AIR proper name detection module utilizes clue words in the document text to indicate the presence of a proper Noun. These proper Noun clues are divided into six different categories: People, Major Cities, Locations, Countries, Organizations and Potential Terrorist Groups. By Locations we mean anything that is not a country or a major city name but yet indicates a location, such as villages or counties. The clues are specific to each category and are hand crafted. Our detection approach can be described as rule-based. The proper name module restricts the output to contain only the exact name of the instance detected.

Clue	proper name Category
الرئيس (president), type = title	People
شركة (company), type = description	Organizations
مدينة (city), type = description	Locations

Table 8. Proper Name clues.

Major cities and countries are detected by comparing the given word from the document against the manually generated list of countries and cities. To detect a person's name, we utilize the "title" of that person and extract the title and the name. A similar procedure applies for the other three categories where we look for the clue word and then output the proper name along with its specifics. These clue words are needed for database purposes where they are further subdivided into subcategories.

The detection procedure starts building the proper name "window" once it encounters a clue word indicating the start of a possible proper name. Every consecutive word that is considered to be part of the proper name is added to the window until the end of the proper name has been reached. The end is recognized by another set of clues consisting of punctuation and additional clue words. For the People category there is an additional step in processing. Here the title is separated from the actual proper name through the use of yet another set of clue words. The potential proper name is discarded if no name was detected after the title.

Our current approach runs best on Arabic newspaper text because news articles tend to provide extra information with their proper name entities that we use as clue words. In absence of these clue words, we do not detect proper names with the same efficiency. At present, the program may not detect proper names that lack these clues. The city names clues currently point to major cities (for example, capital cities) only. As clues to city names are largely formed by lists of city names, it is difficult to detect smaller cities. In future work we hope to extend our city clue lists.

To test our new approach, we randomly selected 120 Arabic documents from the TREC collection. These documents were analyzed manually by a native Arabic speaker to extract all their proper names. These proper names formed the gold standard we tested our system against. The program ran the same 120 documents and it was able to detect 83.2% of the gold standard proper names, with an average of 1.1 false hit per document. We then removed the files related to sports and entertainment from the original test set. The new set contains 104 documents. After rerunning the program, the results showed that the system now detects 86.4% of the gold standard proper names with on average a single false hit per document. A third experiment was conducted where we chose to ignore minor city names, as our program is currently not geared to detect them, and excluded them form the study. The system then detected 92.8% of the gold standard proper names, the number of false hits remaining steady of one per file on average.

Out of the six categories, only the countries and major-cities categories were evaluated. The system achieved a 100% hit rate in both categories. Our evaluation shows that since many of the rules were built using documents about politics, they don't perform as well on documents about sports and entertainment. It is also clear that rules are never completely error proof and false hits are possible under some circumstances. The fact that only 8% of person names are not detected in political related documents shows that the derived rules produced very good results. Our future work will aim to increase the global hit rate by including new rules to accommodate sports and entertainment related files. In addition we will increase the ability of the program to detect the names of minor-cities.

6. Publications

We reported on our work for the AIR project at different venues and will continue to do so in the future.

Diekema, A. R. Preliminary Lexical Framework for English-Arabic Semantic Resource Construction. COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages. Geneva, Switzerland, 2004.

Diekema, Anne R.; Hannouche, Jean; Ingersoll, Grant; Oddy, Robert N. (2004) Arabic Information Retrieval (AIR). Poster presented at Homecoming Weekend, Syracuse, NY, October 9.

Diekema, Anne R., and Liddy, Elizabeth D. (To appear) Analyst-Focused Arabic Information Retrieval. 2005 International Conference on Intelligence Analysis Methods and Tools, McLean, VA. 2-6 May, 2005.

Diekema, A.R. Hannouche, Jean; Ingersoll, Grant; Oddy, Robert N.; and Liddy, E.D. Cross-Language Information Retrieval in Arabic: CNLP's AIR system (2005) (In Press) Brown Bag Lunch Series Spring 2005. Syracuse University, School of Information Studies. Syracuse, NY.

Acknowledgments

This major undertaking has received generous support secured by Senator Charles E. Schumer, administered through the Department of Justice and the Department of Homeland Security.

References

AbdulJaleel, N. and Larkey, L. (2003). Statistical transliteration for English-Arabic Cross Language Information Retrieval. *CIKM 2003: Proceedings of the twelfth international conference on information and knowledge management* , New Orleans, LA, 139-146.

Aljlayl, M. , S. Beitzel, E. Jensen, A. Chowdhury, D. Holmes, M. Lee, D. Grossman, O. Frieder. (2001) ITT at TREC-10. TREC-2001.

Aljlayl, Mohammed and Ophir Frieder. (2001) Effective Arabic-English Cross-Language Information Retrieval via Machine-Readable Dictionaries and Machine Translation. CIKM '01.

Chen, Aitao and Fredric Gey. (2001) Translation Term Weighting and Combining Translation Resources in Cross-Language Retrieval. TREC-2001.

Chen, Aitao and Fredric Gey. (2002) Building an Arabic Stemmer for Information Retrieval. TREC-2002.

Chen, J. (2003) The Construction, Use, and Evaluation of a Lexical Knowledge Base for English-Chinese Cross-Language Information Retrieval. Dissertation. School of Information Studies, Syracuse University.

Chowdhury, Abdur, Mohammed Aljlayl, Eric Jensen, Steve Beitzel, David Grossman, Ophir Frieder. (2002) IIT at TREC-2002: Linear Combinations Based on Document Structure and Varied Stemming for Arabic Retrieval. TREC-2002.

Darwish, K. and D.W. Oard. (2002_2) *CLIR Experiments at Maryland for TREC-2002: Evidence combination for Arabic-English retrieval*. In "Proceedings of the Eleventh Text REtrieval Conference (TREC-11)", E.M. Voorhees and C.P. Buckland ed., pages 703-710, NIST, Gaithersburg, MD.

Darwish, Kareem and Douglas W. Oard. (2002_1)Term Selection for Searching Printed Arabic. SIGIR '02. 261-268.

Darwish, Kareem, David Doermann, Ryan Jones, Douglas Oard, and Mika Rautiainen. (2001) TREC-10 Experiments at University of Maryland CLIR and Video. TREC-2001.

Diekema, Anne. R. (2003). Translation Events in Cross-Language Information Retrieval: Lexical Ambiguity, Lexical Holes, Vocabulary Mismatch, and Correct Translations. Dissertation. School of Information Studies, Syracuse University.

Franz, Martin and J. Scott McCarley. (2002) Arabic Information Retrieval at IBM. TREC-2002.

Fraser, Alexander, Jinxi Xu, and Ralph Weischedel. (2002) TREC 2002 Cross-lingual Retrieval at BBN. TREC-2002.

Gey, Fred C. and Douglas W. Oard. (2001) The TREC-2001 Cross-Language Information Retrieval Track: Searching Arabic using English, French, or Arabic Queries. TREC-2001.[5]

Kwok, K.L. , L. Grunfeld, N. Dinstl, M. Chan. (2001) TREC2001 Question-Answer, Web and Cross Language Experiments using PIRCS. TREC-2001.

[5] All TREC papers available online at http://trec.nist.gov/pubs.html

Larkey, Leah S. and Margaret E. Connell. (2001) Arabic Information Retrieval at UMass in TREC-10. TREC-2001.

Larkey, Leah S., James Allan, Margaret E. Connell, Alvaro Bolivar, and Courtney Wade. (2002) UMass at TREC 2002: Cross Language and Novelty Tracks. TREC-2002.

Larkey, Leah, Lisa Ballesteros, Margaret E. Connell. (2002) Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. SIGIR 2002. 275-282.

Levenshtein, Vladimir I. (1966) *Binary codes capable of correcting deletions, insertions, and reversals*, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian). English translation in Soviet Physics Doklady, 10(8):707-710, 1966.

Levow, Gina-Anne, Douglas W. Oard, and Philip Resnik. Dictionary-Based Techniques for Cross-Language Information Retrieval. [IP&M paper under review]

Mayfield, James, Paul McNamee, Cash Costello, Christine Piatko, Amit Banerjee. (2001) JHU/APL at TREC2001: Experiments in Filtering and in Arabic, Video, and Web Retrieval. TREC-2001.

McNamee, Paul, Christine Piatko, James Mayfield. (2002) JHU/APL at TREC 2002: Experiments in Filtering and Arabic Retrieval. TREC-2002.

Miller, G. (1990). WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4), Special Issue.

Oard, Douglas and Diekema, Anne R. (1998). Cross-Language Information Retrieval. *Annual Review of Information Science*, 33: 223-256.

Oard, Douglas W. and Fredric C. Gey. (2002) The TREC-2002 Arabic/English CLIR Track. TREC-2002.

Pirkola, Ari (1999). Studies on Linguistic Problems and Methods in Text Retrieval: The Effects of Anaphor and Ellipsis Resolution in Proximity Searching, and Translation and Query Structuring Methods in Cross-Language Retrieval. Academic Dissertation. University of Tampere, Department of Information Studies, Finland.

Rocchio, J.J. (1971) Relevance Feedback in Information Retrieval. In: Gerard Salton, Ed. The SMART Retrieval System - Experiments in Automatic Document Processing, Chapter 14, Prentice Hall, Englewood Cliffs, N.J.

Sanderson, Mark and Asaad Alberair. (2001) Keep It Simple Sheffield – a KISS approach to the Arabic track. TREC-2001.

Savoy, Jacques and Yves Rasolofo. (2002) Report on the TREC-11 Experiment: Arabic, Named Page and Topic Distillation Searches. TREC-2002.

Tomlinson, Stephen. (2001) Hummingbird SearchServer at TREC 2001. TREC-2001.

Tomlinson, Stephen. (2002) Experiments in Named Page Finding and Arabic Retrieval with Hummingbird SearchServer at TREC 2002. TREC-2002.

Voorhees, E.M. and Harman, D. (1999). Overview of the Seventh Text REtrieval Conference (TREC-7). In: *NIST Special Publication 500-242: The Seventh Text REtrieval Conference (TREC-7)*. E.M. Voorhees and D.K. Harman, (Eds.). Gaithersburg, MD: Department of Commerce, National Institute for Standards and Technology, 1-24.

Xu, Jinxi, Alexander Fraser, Ralph Weischedel. (2001) TREC 2001 Cross-lingual Retrieval at BBN. TREC-2001.

Xu, Jinxi, Alexander Fraser, Ralph Weischedel. (2002) Empirical Studies in Strategies for Arabic Retrieval. SIGIR '02. 269-274.

APPENDIX 1: AIR USER STUDY HANDOUT

AIR User Study

Thank you for participating in the AIR User Study!

1. Brief introduction to the AIR system

AIR is a cross-language information retrieval (CLIR) system providing English speakers with the ability to search Arabic content without knowledge of Arabic.

Unlike regular search engines, the AIR system is designed to allow the user to track topics of interest over time and organize the results into a comprehensive answer to the user's information need.

Using the AIR system revolves around understanding the relationship between *topics*, *enquiries*, and *searches*.

Simply put, a topic is a broad description of the information need in general. It is used for organizing information over time and is not directly used in the search. An Enquiry is a specific query or question that is related to the Topic.

For example, as an analyst I may be interested in tracking the Muslim Brotherhood. More specifically I may want to know how it spread from Egypt to other countries, and by what other names this organization is known.

Topic:	Muslim Brotherhood
Enquiries:	1) spread of the Muslim Brotherhood spread from Egypt to other countries 2) other names of the Muslim Brotherhood

A Search is an Enquiry that is executed against a specific database.

2. Create a Topic

Please carry out the following tasks:

1) Think of a general topic area relevant to the Middle East.
2) Create your Topic area

3. Create an Enquiry

Please carry out the following tasks:

1) Think of an Enquiry (query) you want to ask the system that fits your previously defined Topic.

2) Create your Enquiry

4. Carry out a search
Please carry out the following task:

1) Carry out a system search with your query.

5. Translation disambiguation

Please carry out the following task:

1) Select the terms to be included in your query.

6. View search results

Please carry out the following task:

1) Select a document or two to look at in detail.

7. Relevance feedback

Please carry out the following tasks:

1) Mark a few relevant and non-relevant documents.
2) Look at the next page of your search results.

8. View relevant results

Please carry out the following tasks:

1) Mark a few relevant and non-relevant documents.
2) Look at the next page of your search results.

9. Debriefing

Please share your thoughts on your AIR system experience.