Arabic Information Retrieval at UMass in TREC-10

Leah S. Larkey and Margaret E. Connell
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA USA 01002
{larkey|connell}@cs.umass.edu

1. Introduction

The University of Massachusetts took on the TREC10 cross-language track with no prior experience with Arabic, and no Arabic speakers among any of our researchers or students. We intended to implement some standard approaches, and to extend a language modeling approach to handle co-occurrences. Given the lack of resources – training data, electronic bilingual dictionaries, and stemmers, and our unfamiliarity with Arabic, we had our hands full carrying out some standard approaches to monolingual and cross-language Arabic retrieval, and did not submit any runs based on novel approaches.

We submitted three monolingual runs and one cross-language run. We first describe the models, techniques, and resources we used, then we describe each run in detail. Our official runs performed moderately well, in the second tier (3rd or 4th place). Since submitting these results, we have improved normalization and stemming, improved dictionary construction, expanded Arabic queries, improved estimation and smoothing in language models, and added combination of evidence, increasing performance by a substantial amount.

2. Information Retrieval Engines

We used INQUERY [2] for two of our three monolingual runs and our cross-language run, and language modeling (LM) for one monolingual run. The processing was carried out using in-house software which implemented both engines, to insure that the stop lists, tokenization, and other details were identical. The same tokenization was used in indexing the Arabic corpus and processing Arabic queries. In fact, except for one minor difference in tokenization, Arabic strings were treated exactly like English strings – as a simple string of bytes, regardless of how they would be rendered on the screen. For both English and Arabic, text was broken up into words at any white space or punctuation characters. The minor difference in Arabic tokenization consisted of five additional Arabic punctuation characters included in the definition of punctuation. Words of one-byte length (in CP1256 encoding) were not indexed.

2.1. Inquery

Two of the three monolingual runs and the cross-language run used a version of INQUERY as the search engine. This version computes the belief function reported in UMass's TREC9 report [1]. The main difference between this version and "real" INQUERY is that proximity information is not stored in the index, so that INQUERY operators requiring proximity information are not implemented.

2.2. Language Modelling (LM)

2.2.1. Monolingual

In language modeling, documents are represented as probability distributions over a vocabulary. Documents are ranked by the probability of generating the query by randomly sampling the document model. The language models here are simple unigram models, similar to those of [7] and [9]. Unigram

probabilities in our official run were estimated as a mixture of maximum likelihood probability estimates from the document and the corpus, as follows:

$$P(Q \mid Doc) = \prod_{q \in Q} \left(\lambda P(q \mid Doc) + (1 - \lambda) P(q \mid BG) \right)$$

where *P(Q/Doc)* is the probability of generating the query from the document model, *q* are the words in the query, λ is a mixture parameter, *P(q/BG)* is the probability of the query word in the background model, and *P(q/DOC)* is the probability of the query word in the document. Normally, the maximum likelihood probabilities are estimated as:

$$P(q \mid Doc) = \frac{tf_{q,Doc}}{|Doc|}$$

where $tf_{q,Doc}$ is the number of occurrences of term q in document, and $|Doc|$ is the length of document, that is, the number of total term occurrences in the document. In an analogous manner, the background probabilities are estimated from a collection *C* which may or may not be the collection in which the document resides, as:

$$P(q \mid BG) = \frac{tf_{q,C}}{|C|}$$

where $tf_{q,C}$ is the number of occurrences of term q in the collection *C*, and $|C|$ is the number of total occurrences of all terms in *C*.

For our official run, we estimated background probabilities as above, and we estimated λ via the Witten Bell method [10], in which

$$\lambda = \frac{|Doc|}{|Doc| + N_{Doc}}$$ where N_{Doc} is the number of different terms in the document.

Posthoc work on Arabic and other data has shown improvements in monolingual LM retrieval by modifying how λ, the mixture parameter, is calculated. For long (expanded) queries, we set λ to a constant value of .4. For short (unexpanded) queries we use Dirichlet smoothing [11], that is,

$$\lambda = \frac{|Doc|}{|Doc| + k}$$ where $k = 800$.

We have also found better LM performance on Arabic and other data if we use document frequencies rather than term frequencies for background models, as in [3], that is:

$$P(q \mid BG) = \frac{df_{q,C}}{\sum_{t \in C} df_{t,C}}$$ where $df_{q,C}$ is the number of documents in *C* containing term *q*, and the summation is over all the terms in the collection.

3. The Arabic Corpus

The AFP ARB corpus of 383,872 documents in Arabic was converted to CP1256 encoding and normalized in the manner described above. The corpus was indexed in two different ways. For the non-stemmed conditions (UMass4), the corpus was normalized, tokenized using the Arabic tokenizer, and every token longer than one byte in length was indexed. For the stemmed conditions (UMass1, UMass2, and UMass3), stemmed tokens longer than one byte in length were indexed.

4. Arabic Resources and Techniques

4.1. Normalization of Arabic

In order to handle the variations in the way text can be represented in Arabic, we performed several kinds of normalization on text in the corpus and in the queries. The normalized form of the corpus was used for indexing (in the non-stemmed conditions), and queries were normalized before they were submitted to the search engine. Dictionaries were also normalized, so that their output would match the forms found in the queries and the corpus.

In our official runs, normalization consisted of the following steps:

- Convert to Windows Arabic encoding (CP1256), if necessary
- Remove punctuation
- Remove diacritics (mainly weak vowels) Most of the corpus did not contain weak vowels. Some of the dictionary entries contained weak vowels. This made everything consistent.
- Remove non letters
- Replace initial إ or أ with bare alif ا.
- Replace آ with ا
- Replace the sequence ىء with ئ
- Replace final ى with ي
- Replace final ة with ه

The definitions of punctuation, diacritics, and non-letters came from the Khoja stemmer, below.

We later improved normalization substantially via two minor changes – replacing إ or أ with bare alif ا regardless of position in the word, and removing tatweel. The label *norm* refers to the original normalization. *Norm2* refers to the modified normalization, and includes stop word removal.

4.2. Stemming

We obtained a stemmer from Shereen Khoja of the Computing Department at Lancaster University [4], which we modified to suit our needs. The stemmer included several useful data files such as a list of all diacritic characters, punctuation characters, definite articles, and 168 stop words, etc. We used some of these files in our normalization algorithm above. This stemmer attempts to find roots for Arabic words, which are far more abstract than stems. It first removes definite articles, prefixes, and suffixes, then attempts to find the root for the stripped form. If no root is found, then the word is left intact. The stemmer also removes stop words. We know both that roots are too abstract for effective information retrieval, and that the overall approach of not stripping any affixes at all is faulty. Although this stemmer made many mistakes, it improved performance immensely, nevertheless.

The changes we made to the Khoja stemmer were (1) If a root were not found, the normalized form was returned, rather than returning the original unmodified word. (2) We added the list of place names described in section 4.3.4 as "unbreakable" words exempt from stemming.

In addition to the Arabic stop word list included in the Khoja stemmer, we applied a script to remove stop phrases, which were translations of the stop phrases we had in our English stop-phrase removal script.

After TREC we developed a light stemmer which strips definite articles (ال، وال، بال، كال، فال) and و (*and*) from the beginnings of normalized words and strips 10 suffixes from the ends of words (ها، ان، ات، ون، ين، يه، ية، ه، ة، ي) [6]. With stop word removal this stemmer yielded higher performance than the khoja stemmer. In the Results sections below, *khoja* refers to the original Khoja stemmer, *khoja-u* refers to the version where words on city and country list are considered unbreakable and exempt from stemming. *Light* refers to the light stemmer.

4.3. Dictionaries

Our structural approach to query translation for cross-language retrieval required that we look up each individual English word in each query (including words added by query expansion), and get all available translations into Arabic words or phrases. We put together several different sources of translations for English words into Arabic, using free resources from the web as much as possible.

4.3.1. The Ectaco dictionary

The Ectaco dictionary is available online, at `http://www.ectaco.com/online`. We could not query this dictionary under program control, so we collected entries manually from the web site. For each English query term and expanded query term, we collected entries by cutting and pasting all the Arabic translations that were available. If an English word were not found, we searched for the word as stemmed by the *kstem* stemmer.

4.3.2. The Sakhr multilingual dictionary

The Sakhr multilingual dictionary is found at `http://dictionary.ajeeb.com/en.htm`. We were able to harvest entries from this dictionary under the control of a Java program which repeatedly queried the English to Arabic page with English words. We collected all available definitions for query words and expansion words. In addition, we collected Arabic-English entries for all available Arabic words in the AFP_ARB corpus.

4.3.3. Sakhr SET machine translation

The Sakhr SET machine translation engine was made available to TREC participants by Mark Meinke at `http://217.52.128.36/set/English/`. This was not used to translate queries. We used it only to look up individual words that we did not find in either of the two dictionaries. This MT engine has a transliteration component, which converts the English word into Arabic characters if a translation is not found. We used this as a substitute for a transliteration algorithm, which we did not yet have available.

4.3.4. Place name lexicon

A small bilingual lexicon of country and city names was derived from a list of world cities we found on the web at `http://www.fourmilab.ch/earthview/cities.html`. This list had 489 entries, and listed the names of most countries of the world, their capitals, and a few other major cities. To get the Arabic translations, we used the Sakhr SET engine, which performed machine translation from English to Arabic. Many of these translations were transliterations. This list of place names (and only this list, which was made independently of the queries) was hand corrected by an Arabic speaking consultant.

4.3.5. Small and large lexicons

Two bilingual lexicons were built. The first (*small*) consisted of the place names plus all the English-Arabic translations found for all of the English query words, including the additional query words added via expansion of the English query for the cross-language run. The second (*large*) lexicon consisted of all the entries from the small lexicon, plus the all the inverted Arabic-English entries. For convenience, we built stemmed versions of the lexicons for each stemmer that we tested. The small normalized English to Arabic lexicon contained 28,868 English words, 269,526 different Arabic translations, for an average of 9.3 different translations per word. The large normalized lexicon contained 50,358 English words, 1,692,408 translations, for an average of 33.6 different translations per word.

5. Other Resources and Techniques

5.1. Stop words and phrases

English stop words (used only for cross-language retrieval) are from INQUERY's standard list of 418 stop words. English stop phrases are defined by regular expressions in a script we have used before in TREC (in English). We built a list of Arabic stop phrases from this by translating the phrases. Arabic stop words are from the Khoja stemmer's list of 168 stop words.

5.2. Query Expansion

We expanded English queries in our official cross-language run, using the AP news articles from 1994 through 1998 in the Linguistic Data Consortium's NA News corpus. This corpus was indexed without stemming, but normalized to lower case. We retrieved the top 20 documents for each query, and ranked the terms from these documents using the ratio method described in Ponte's thesis, chapter 5 [8]. The five top ranked new English terms were then added to the query. Each term in the query received a final weight of $2w_o + w_e$ where w_o is the original weight in the unexpanded query, and w_e is the score the term received by the ratio method expansion.

After submitting the official runs, we changed the expansion method. Terms from the top 10 documents received an expansion score which was the sum across the ten documents of the Inquery belief score for the term in the document. The 5 terms with the highest expansion score were added to the query. Final weights were set to $2w_o + w_e$ where w_o is the original weight in the unexpanded query and w_e=1.

Due to technical problems involving the interaction of Arabic stemming with query expansion, and lack of time we did not submit any official runs in which the Arabic queries (monolingual, or translated for cross-language) had been expanded.

After TREC, we added Arabic query expansion, performed as follows: retrieve the top 10 documents for the Arabic query, using LM retrieval if the expanded query would be run in an LM condition, and using Inquery retrieval if the expanded query would run in an Inquery condition. Terms from the top ten documents were ranked using the same expansion score used in the post-hoc English expansion. The top 50 terms that were not already part of the original query were added. For Inquery conditions, the added terms were added to the original query as additional terms under the top level #wsum operator. For both Inquery and LM conditions, the weights on original terms were doubled, and the new terms received a weight of 1.

6. Monolingual Runs

6.1. Description of Runs

We entered three monolingual runs, which differed in stemming and in retrieval algorithms:

1. **Inquery Baseline**. (UMass4) : normalized but not stemmed
2. **Inquery Stemmed** (UMass1): stemmed using Khoja-u stemmer
3. **LM Stemmed** (UMass2): stemmed using Khoja-u stemmer. LM as described in section 2.2.1.

The following steps were carried out in processing all monolingual runs.

1. Convert queries to CP1256 encoding.
2. Remove all but the title and description fields.
3. Remove stop phrases from Arabic queries.

4. Normalize or stem the query, depending on the condition.
5. Rank the documents using either INQUERY or LM, depending on condition.

6.2. Results

Without stemming our system performed very poorly. With stemming it performed quite well, as summarized in Table 1. The table shows average precision for each run, and summarizes a query-by-query comparison with the median performance over 20 monolingual manual and automatic runs, with respect to average precision and the number of relevant documents returned in the top 1000.

As the table shows, stemming improves the results immensely. With stemming, average precision improved 49% over the INQUERY baseline. The LM stemmed condition was not as good as the Inquery stemmed condition. A striking pattern apparent in the table is a recall bias due to stemming. In both stemmed conditions the number of queries above the median in relevant documents returned in the top 1000 is larger than the number of queries above the median in average precision.

Table 1: Monolingual Results - official runs with normalization and khoja-u stemmer

CONDITION	Name of Run	Average Precision	Number of Queries at or Above Median	
			Average Precision	Rel Ret in top 1000
Inquery baseline	UMass4	.2104	10/25	10/25
Inquery stemmed	UMass1	.3129	18/25	24/25
LM baseline	not submitted	.1858		
LM stemmed	UMass2	.2597	16/25	20/25

6.3. Posthoc Monolingual Experiments

We compare the official results with runs using improved normalization, stemming, and with query expansion, and better language modeling. Table 1 shows the old and new conditions including the official runs, which are asterisked. *Raw* means that no stemming or stop word removal was applied. *Norm, norm2, khoja-u, khoja,* and *light* are defined in section 4.2 above. Since roots and lightly stemmed words are quite different representations of Arabic words, we reasoned that they could be productively combined. *Light+khoja* is a combination of evidence run, where the ranked lists from the light and khoja runs were averaged without any normalization of scores. Shaded cells were conditions that were not run.

Table 2: Monolingual results with improved normalization, stemming, and language modeling, with and without query expansion

	Raw	Norm	Norm2	Khoja-u	Khoja	Light	Light+ Khoja
Inquery	.1935	**.2104***	.2408	**.3129***	.3410	.3894	.4088
Inquery + Query Expansion	.2709	.3002	.3303	.3595	.3778	.4274	.4408
LM		.1858		**.2597***			
LMnew	.1879	.2020	.2431	.3189	.3479	.3736	.3981
LMnew+Query Expansion	.2629	.2990	.3335	.3490	.3772	.4130	.4465

* official runs

It is apparent from these runs that the light stemmer is superior to the khoja stemmer. Although it seemed like a good idea to have the list of unbreakable place names as part of the Khoja stemmer, performance

was better without it. These results also show that the changes in background model estimation and smoothing bring language model performance to a level comparable to that of Inquery.

7. Cross Language Retrieval

7.1. Description

Our official cross language run (UMass3) used the INQUERY search engine, the Khoja stemmer (with unbreakables) for Arabic, the *kstem* stemmer for English [5], and query expansion of the English query, dictionary lookup of query terms in the small dictionary. The steps were as follows:

1. Remove stop phrases from English queries.
2. Remove stop words from English queries
3. Expand the English query
4. For each English word:
    a. Look for a set of translations in all of the English to Arabic lexicons described above
    b. If not found, stem the English word using the *kstem* stemmer and look it up again. Use all translations found in the dictionary.
    c. Stem the Arabic translations
    d. If any of the translations consist of an Arabic phrase rather than a single word, enclose the phrase in a **`#filreq`** operator. **`#filreq`** is like a Boolean *and*. If this version of INQUERY had proximity information, we would have used phrase or ordered window operators instead, but these were not available.
    e. If a set of translations was found, enclose all the alternatives in a **`#syn`** (synonym) operator
5. Build a weighted sum query out of all the stemmed translations of the query terms by subsumimg all the synonym sets under a **`#wsum`** (weighted sum) operator. Each synonym set was given the weight described above in the query expansion section.
6. Submit the weighted sum query to Inquery to retrieve Arabic documents.

7.2. Results

Table 3: Cross Language Results – official run – Inquery, expanded English, unexpanded Arabic

CONDITION	Name of Run	Average Precision	Number of Queries at or Above Median	
			Average Precision	Rel Ret in top 1000
Inquery baseline	not submitted	.1691		
Inquery stemmed	UMass3	.2795	20/25	20/25

Table 3 shows the results for the Cross Language run in the same format as the Table 1. In this case, query-by-query performance is compared with the median of 28 cross language runs, which include 2 French to Arabic, and 1 manual run. In 20 out of 25 queries, we performed at or above the median in both average precision and in the number of relevant documents returned in the top 1000.

Subsequent experiments showed improved results using the same general approach, but with the light stemmer, the large dictionary, and Arabic query expansion as well as English.

We compared the small and large dictionaries, described in Section 4.3.5.

Table 4: Comparison of small and large English-to-Arabic lexicons. Unexpanded cross-language retrieval

	norm	**khoja-u**	**light8**
Small lexicon	.1660	.2069	.3655
Large lexicon	.2624	.2514	.3794

Table 4 shows that the large dictionary performed substantially better than the smaller dictionary, in spite of the large number of translations for each word in the large dictionary.

The final set of experiments, summarized in Table 5, show that expanding both English and Arabic queries with the large dictionary and the light8 stemmer give the most effective cross-language retrieval. *Raw* means that no normalization or stemming were applied, *norm*, *khoja-u*, *khoja*, and *light* conditions refer to normalization only, Khoja stemmer with unbreakables, Khoja stemmer without unbreakables, and light stemming, respectively. *Light+khoja* is a combination of evidence run, in which scores from the *light* and *khoja* runs were averaged. Combination of evidence improves performance, but only slightly.

Table 5: Cross-language retrieval using large lexicon, different stemmers, and query expansion

	raw	**norm**	**khoja-u**	**khoja**	**light**	**light+khoja**
No query expansion	.1128	.2624	.2514	.2598	.3794	.3830
Expanded English	.1389	.3056	.2934	.3077	.4222	.4348
Expanded Arabic	.1544	.3371	.2917	.2931	.4106	.4189
Expanded English and Arabic	.1690	.3480	.3516	.3589	.4502	.4629

8. Acknowledgments

We would like to thank Shereen Khoja for providing her stemmer, Nicholas J. DuFresne for writing some of the stemming and dictionary code, Fang-fang Feng for helping with dictionary collection over the web, Mohamed Taha Mohamed , Mohamed Elgadi, and Nasreen Abdul-Jaleel for help with the Arabic language, Victor Lavrenko for the use of his vector and language modeling code and his advice.
This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

9. References

[1] Allan, J., Connell, M. E., Croft, W. B., Feng, F.-f., Fisher, D., and Li, X.INQUERY and TREC-9. presented at The Ninth Text REtrieval Conference (TREC-9), Gaithersburg, Maryland, 2000.

[2] Callan, J. P., Croft, W. B., and Broglio, J. TREC and TIPSTER Experiments with INQUERY. *Information Processing and Management*, 31 (3), pp. 327-343, 1995.

[3] Hiemstra, D. and de Vries, A. *Relating the new language models of information retrieval to the traditional retrieval models*. University of Twente, Enschede, The Netherlands, CTIT Technical Report TR-CTIT-00-09, May 2000.

[4] Khoja, S. and Garside, R. *Stemming Arabic Text*. Computing Department, Lancaster University, Lancaster, U.K. http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps, September 22, 1999.

[5] Krovetz, R. Viewing Morphology as an Inference Process. in *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993, pp. 191-203.

[6] Larkey, L. S., Ballesteros, L., and Connell, M. E. *Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis*. University of Massachusetts, Amherst, MA, CIIR Technical Report IR-249, 2002, submitted to SIGIR 2002.

[7] Miller, D. R. H., Leek, T., and Schwartz, R. M. A Hidden Markov Model Information Retrieval System. in *Proceedings of SIGIR '99: 22nd International Conference on Research and Development in Information Retrieval*. Berkeley: ACM Press, 1999, pp. 214-221.

[8] Ponte, J. M. A Language Modeling Approach to Information Retrieval. Dissertation, Department of Computer Science. Amherst, MA: University of Massachusetts, 1998, pp. 158 pages.

[9] Song, F. and Croft, W. B. A general language model for information retrieval. in *Proceedings of the Eighth International Conference on Information Knowledge Management (CIKM '99)*: ACM Press, 1999, pp. 316-321.

[10] Witten, I. H. and Bell, T. C. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37, pp. 1085-1094, 1991.

[11] Zhai, C. and Lafferty, J. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. in *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval.* New Orleans: ACM Press, 2001, pp. 334-342.