Effective Retrieval Techniques for Arabic Text

A thesis submitted for the degree of
Doctor of Philosophy

Abdusalam F Ahmed Nwesri
B.Sci., M.Soft.Eng.,

School of Computer Science and Information Technology
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

May, 2008

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Abdusalam F Ahmed Nwesri
School of Computer Science and Information Technology
RMIT University
May, 2008

Acknowledgments

First, I thank Allah the Almighty for giving me the strength and the ability to complete this thesis. Many thanks to my parents who are patiently waiting for me to come home to support them for the rest of their life. I am also grateful to my wife and family for their support, understanding and constant encouragements.

I thank Saied Tahaghoghi and Falk Scholer for their support and patience. Without their guidance and advice this thesis would not exist. My thanks go to Hugh Williams for his initial support when I first started this thesis and to Justin Zobel for his comments on Chapter 7.

Many thanks to those contributed to this research either directly or indirectly. I warmly thank those participated in annotating the AGW corpus: Ahmad Omran, Ashraf Gadri, Anwar Al-Eisawy, Aiman Attarhony, Rabee Swisse, Mohamed Abushhiwa, Abdulkareem Elbaz, Mansor Moftah, Abdul-Fatah Khorwat, Abdurrahman Ertep, Abdurrazag Mezughi, Miluod Asarat, Salem Aburrema, Khaled Abdulgader, Eltaher alshagamany, Abdulmajeed Abaza, Redha Omran, Maher bin Abdul-Muhsen, Bushra Zawaydeh, and Abdul-Minem Sanallah. I thank Timo Volkmer for his help in aligning ASR text with video shots in the TRECVID 2005 collection. I also thank those who participated in reading my thesis: Wigdan Mohamed, Abeer Ajlouni, Yussuf Hart, Philip Crooks, Fawziya Abderrahim, Rafig Annabulsi, Yohannes Tsegay, Nadim Rafehi, Wasim Wardak, and Fatmir Badali.

For four years, I was part of the Search Engine Group at RMIT University. I would like to thank all the members of the group specifically, Jelita Asian, Sarvnaz Karimi, Yohannes Tsegay, Iman Suyoto, Dayang Iskandar, Steven Burrows, Jonathan Yu, Nikolas Askitis, and Halil Ali.

I also want to thank James Thom for giving me feedback on this thesis; Chin Scott, Beti Dimitrievska, and Nyree Freeman for their support as research programs administrator at our school.

I would like to thank Microsoft Corporation for providing me with a copy of the Microsoft Office Proofing Tools 2003. This research is supported by the Libyan government scholarship through its Bureau in Canberra. Many thanks go to them.

Credits

Portions of the material in this thesis have previously appeared in the following publications:

- Abdusalam F.A. Nwesri, S.M.M. Tahaghoghi, and Falk Scholer. Stemming Arabic conjunctions and prepositions. In M. Consens and G. Navarro, editors, Proceedings of String Processing and Information Retrieval, 12th International Conference, SPIRE 2005, pages 206–217, Buenos Aires, Argentina, 2–4 November 2005. Springer, Heidelberg, Germany. ISBN 3-540-29740-5.

- Abdusalam F.A. Nwesri, S.M.M. Tahaghoghi, and Falk Scholer. Capturing out-of-vocabulary words in Arabic text. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pages 258–266, Sydney, Australia, 22–23 July 2006. Association for Computational Linguistics.

- Abdusalam F.A. Nwesri, S.M.M. Tahaghoghi, and Falk Scholer. Arabic text processing for indexing and retrieval. In Proceedings of the International Colloquium on Arabic Language Processing, Rabat, Morocco, 18–19 June 2007. In Arabic.

- Abdusalam F.A. Nwesri, S.M.M. Tahaghoghi, and Falk Scholer. Finding variants of out-of-vocabulary words in Arabic. In Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, pages 49–56, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

- Abdusalam F.A. Nwesri, S.M.M. Tahaghoghi, and Falk Scholer. Answering English queries in automatically transcribed Arabic speech. In Proceedings of the 6th Annual IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), pages 11–16. IEEE Computer Society, 11–13 July 2007.

The thesis was written in the `WinEdt 5.5` editor on Windows 2000, and typeset using the `MiKTeX 2.5` document preparation system.
All Arabic scripts are written using the ArabTeX package written by Klaus Lagally at the Institut fuer Formale Konzepte der Informatik, University of Stuttgart, Stuttgart, Germany. All transliterations are represented using the International Phonetic Association (IPA) conventions.

Contents

List of Figures

List of Tables

Abstract

Arabic is a major international language, spoken in more than 23 countries, and the lingua franca of the Islamic world. The number of Arabic-speaking Internet users has grown over nine-fold in the Middle East between the year 2000 and 2007, yet research in Arabic Information Retrieval (AIR) has not advanced as in other languages such as English. Most techniques used by most current search engines are still limited to the use of word as a search unit, despite the fact that Arabic is a highly inflected language. In this thesis, we explore techniques that improve the performance of AIR systems.

Stemming is the process of reducing words to their roots or stems. In highly inflected languages such as Arabic, stemming is considered one of the most important factors to improve retrieval effectiveness of AIR systems. Most current stemmers remove affixes without checking whether the removed letters are actually affixes. We propose lexicon-based improvements to light stemming that distinguish core letters from proper Arabic affixes. We devise rules to stem most affixes and show the effects of each individual rule on retrieval effectiveness as well as using all rules together. Using the TREC 2001 test collection, we show that applying relevance feedback with our rules produces significantly better results than light stemming.

Techniques for Arabic information retrieval have been studied in depth on clean collections of newswire dispatches. However, the effectiveness of such techniques is not known on other noisy collections such as transcribed news collections in which text is generated using automatic speech recognition (ASR) systems and queries are generated using machine translations (MT). Using noisy collections, we show that normalisation, stopping and light stemming improve results as in normal text collections but that n-grams and root stemming decrease performance.

Test collections play a major role in evaluating alternative IR approaches. Most recent AIR research has been undertaken using collections that are far smaller than the collections used for English text retrieval; consequently, the significance of some published results is

debatable. Using the LDC Arabic GigaWord collection that contains more than 1 500 000 documents, we create a test collection of 90 topics with their relevance judgements. We use this test collection to test the effectiveness of several techniques including our lexicon-based light stemming, and show empirically that for a large collection, root stemming is not competitive. Of the approaches we have studied, lexicon-based stemming approaches perform better than light stemming approaches alone.

Arabic text commonly includes foreign words transliterated into Arabic characters. Several transliterated forms may be in common use for a single foreign word, but users rarely use more than one variant during search tasks. We explore two issues in this area: identification, and retrieval.

We test the effectiveness of lexicons, Arabic patterns, and n-grams in distinguishing foreign words from native Arabic words. We introduce rules that help filter foreign words and improve the n-gram approach used in language identification by determining the best n-grams size to construct word and language profiles. Our combined n-grams and lexicon approach successfully identifies 80% of all foreign words with a precision of 93%.

To find variants of a specific foreign word, we apply phonetic and string similarity techniques and introduce novel algorithms to normalise them in Arabic text. We modify phonetic techniques used for English to suit the Arabic language, and compare several techniques to determine their effectiveness in finding foreign word variants. We test the effectiveness of using such techniques in AIR systems, and show that our algorithms significantly improve recall. We also show that expanding queries using variants identified by our Soutex4 phonetic algorithm results in a significant improvement in precision and recall.

Together, the approaches described in this thesis represent an important step towards realising highly effective retrieval of Arabic text.

Chapter 1

Introduction

The Web has become a major source of information, with billions of documents available for search and more added daily. According to the Netcraft April 2008 survey, there are more than 165 000 000 distinct domain names on the Internet.[1] Given the volume of information available, users increasingly rely on search and filtering tools to find the information they require.

Search engines provide an interface through which people can find information easily in a text collection such as the Web. They collect and index information and employ various techniques to find documents relevant to a user's query.

Few search engines were initially available that support searching documents written in non-Latin characters. However, with the rapid growth of computer use in non-English-speaking regions, search engines have gradually added support for other languages.

While the number of the Internet users in the Middle East increased by 920% between 2000 and 2007, the number of Internet users who use the Arabic language reached 46 359 140 by November 2007, indicating a 1 575.9% growth over the year 2000 (see Figure 1.1).[2]

Search engines employ techniques such as term matching and document ranking that work across most languages. However, application of techniques specific to a target language can help retrieval effectiveness.

Arabic is a highly inflected language. Its words are derived from root words and extended with prefixes, infixes and suffixes; resulting in as many as 2 552 different versions for a verb word and as many as 519 versions for a noun [Attia, 2006]. Moreover, unlike English, prefixes and suffixes in Arabic include pronouns, prepositions, and conjunctions. Most of these affixes

[1] http://news.netcraft.com/archives/2008/04/14/april_2008_web_server_survey.html

[2] http://www.internetworldstats.com/stats7.htm

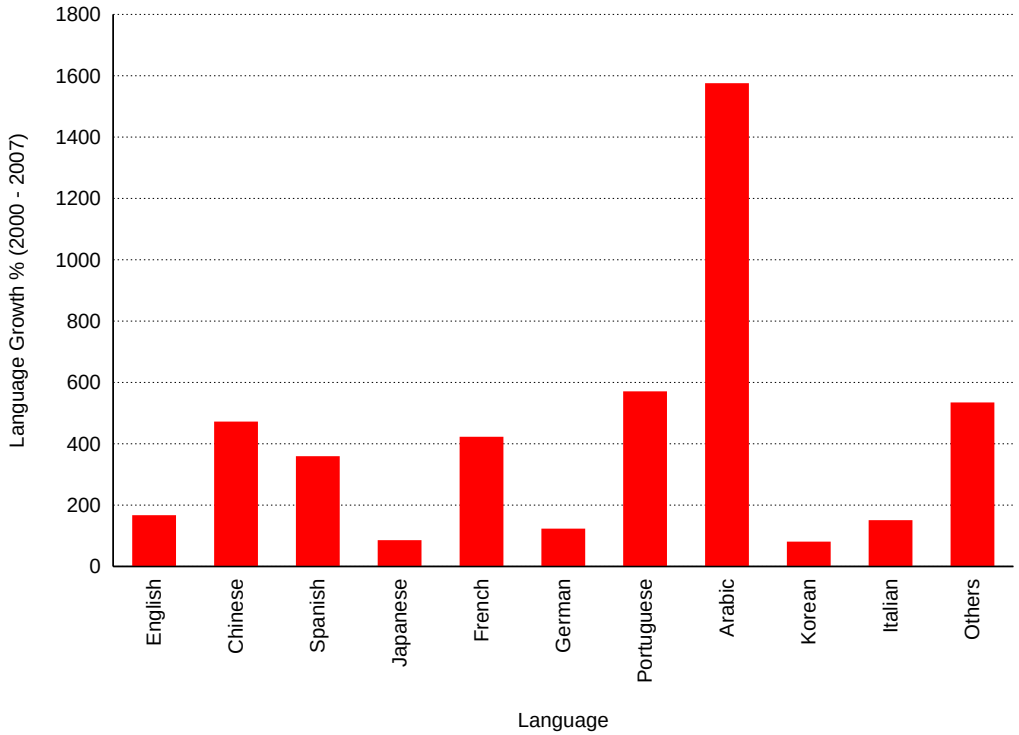

Figure 1.1: Language growth in the Internet between 2000 and 2007. Source: Internet World Stats [Miniwatts International, 2007].

do not appear in a separate form as in English, but are attached to the word. Identification of these affixes is complicated by the fact that they are sometimes identical to core characters of Arabic words. The simple truncation techniques that can be applied to remove English suffixes cannot be used for Arabic, and so we must devise new identification approaches.

Arabic is a highly inflected language; however, even with the rapid increase in Arabic-speaking users, most search engines limit their searches to non-inflected, surface forms of words. Whilst a few dedicated Arabic search engines such as Ayna[3] and Araby[4] are available, most users still rely on major search engines such as Google,[5] Live Search,[6] and Yahoo[7]. Wheeler [2004] surveyed 200 Internet users in Jordan and found that Google was the most

[3]http://www.ayna.com/
[4]http://www.araby.com/
[5]http://google.com
[6]http://live.com
[7]http://yahoo.com

commonly visited web site. However, in another survey, satisfaction rates amongst 26 Arabic users using Google was found to be as low as 32% [Al-Maskari et al., 2007].

Research on Arabic Information Retrieval (AIR) started in earnest in the early 1990s with a few small experiments carried out using small collections of text. Most studies focused on extracting roots and comparing the effectiveness of indexing Arabic text using roots, stems or words. It was only in 2001 that the Text REtrieval Conference (TREC) first dedicated a track to test the effectiveness of techniques in Arabic monolingual and cross-lingual retrieval. Despite the fact that TREC has boosted AIR research, the underlying text collection used for testing is relatively small compared with those used in English. The effectiveness of AIR systems has only been tested using 75 queries which cover a tiny proportion of all Arabic terms.

Compounding these difficulties is the fact that Arabic is a living language which regularly acquires new words from other languages. Such words are problematic in that they do not follow normal Arabic word structure, with many different versions of the same word being used by different people. Existing search engines look for the version submitted in the user query and do not attempt to find other variants in their text collection.

Motivated by the need to enhance monolingual Arabic searching, we investigate techniques that improve AIR effectiveness. We test supporting light stemming with morphological and grammatical rules to avoid removing core letters; test whether techniques used for clean newswire text can also aid retrieval effectiveness for a different collection of automatically transcribed TV news soundtrack; build ground truth for a larger document collection and use it to evaluate existing AIR systems, including the ability to identify foreign words within Arabic text, and the effects of normalising such words on retrieval effectiveness. Specifically, we aim to address the following questions in this thesis.

1.1 How reliable is light stemming with morphological rules?

Arabic is a derivational language in which words are derived from roots and inflected using prefixes, infixes, and suffixes. Versions derived from the same root or stem and sharing the same meaning should be grouped together in one class in the search index. Stemming is the process of returning common inflected words to their stem or root, usually by removing affixes. Removing affixes correctly results in a proper conflation. However, affixes are not distinct in most languages, and stemming often results in an incorrect stem due to mistakes in removing affixes. Such errors conflate incompatible words together in the index, resulting in over-

Before Stemming			After Stemming		
وسام	/wisaam/	⟨proper noun⟩	سام	/saam/	⟨poisonous⟩
الـهام	/ʔlhaam/	⟨inspiration⟩	هام	/haam/	⟨important⟩
بالـتيمور	/balt̪jmwr/	⟨Baltimore⟩	تيمور	/t̪jmwr/	⟨East Timor⟩
عدنـان	/ʕd̪nan/	⟨Adnan⟩	عدن	/ʕd̪n/	⟨Aden port in Yemen⟩

Table 1.1: Effects of light stemming on Arabic words. Affixes are highlighted in red. Light stemmers remove such affixes without validation resulting in another stems with different meanings.

stemming [Paice, 1996]. Despite stemming mistakes, it has been empirically demonstrated that stemming improves retrieval in many languages [Hull, 1996; Popovič and Willett, 1992; Savoy, 1999; Asian, 2007], including Arabic [Larkey et al., 2002; Aljlayl, 2002].

Many experiments have been carried out to test the effectiveness of indexing Arabic text using roots, stems and words. Early experiments have shown that using root forms as the index terms is superior to using stems or the unstemmed words [Al-Kharashi, 1991; Al-Kharashi and Evens, 1994; Abu-Salem, 1992; Abu-Salem et al., 1999; Hmeidi et al., 1997]. These experiments have been carried out on a small collection of abstracts and short documents using manually judged stems and roots.

Automated stemming approaches return words to their stems by removing letters that correspond to Arabic prefixes or suffixes (this is known as light stemming), and may further return stems to roots using patterns (root stemming). Such automatic techniques often fail to produce the exact stem due to ambiguity in Arabic text and due to similarity between affixes and core letters in Arabic words. Light stemmers remove a pre-prepared list of prefixes and suffixes. They compare initial and ending letters of Arabic words with their list and remove matching sequences that pass possible additional criteria, such as that the remaining string should contain at least three characters. Despite the fact that this approach results in many wrong stems (see Table 1.1), it is efficient and improves retrieval effectiveness significantly [Aljlayl and Frieder, 2002; Darwish and Oard, 2003b; Larkey et al., 2007]. In contrast, morphological analysers use lexicons and morphological rules to remove the proper affixes. They analyse all possible combinations of initial and final letters of a word and use rules to validate the combination between these letters and the remaining stem for a given word. Whilst such systems produce more accurate stems, they commonly return more than one possible stem for the same word, making it very difficult to determine the best stem

that represents the word. Such systems are also less efficient than light stemmers, despite sometimes returning results very similar to those of light stemmers [Larkey et al., 2007].

In this thesis, we use an approach that combines light stemming and morphology to produce more effective and efficient results. We compare the effectiveness of existing AIR systems and show that the light stemming techniques are superior to existing systems. We use the light10 stemmer developed by [Larkey and Connell, 2005] as our underlying framework to test the effects of several techniques we propose to remove proper affixes and avoid core letters. Our rules use morphological rules and an Arabic lexicon. We verify whether letters constitute an affix not only by checking whether the word with and without that affix exists in our lexicon, but also by replacing that affix with other equivalent ones and checking the new instances against the lexicon. Our final stemmers achieve results comparable to those of the light10 stemmer, but significantly exceed them when used together with relevance feedback; notably, our lexicon-based stemmers that use the unique terms as a underlying lexicon are three times more efficient than the best morphological analyser.

We explore whether techniques developed on clean data also apply to noisy collections. Using a collection of text generated through automatic speech recognition of TV news soundtrack, and machine translations of English queries, we show that most AIR techniques, with the exception of root-word indexing and n-grams, also apply to the new collection.

1.2 What are the effects of corpus size on the performance of AIR systems?

Test collections play a core role in improving IR systems, as they allow different strategies to be tested. For Arabic, the few available test collections are small compared to those used for English. For example, while the biggest test collections developed for Arabic — the TREC 2001 and TREC 2002 text collections — contain only 383 872 documents (some 800MB of data), the English TREC WT10g collection contains 1.6 million documents (10GB of data), and the English TREC GOV2 text collection contains 25 million documents (420GB of data).

Published results of experiments on small document collections indicate that indexing collections using the word roots is more effective than indexing the stems, or the unstemmed words themselves [Al-Kharashi, 1991; Al-Kharashi and Evens, 1994; Abu-Salem, 1992; Abu-Salem et al., 1999; Hmeidi et al., 1997]. However, other published research that uses the mid-sized TREC 2001 collection shows conflicting results [Aljlayl and Frieder, 2002; Darwish and Oard, 2003b; Larkey et al., 2007]. Using Arabic GigaWord Second Edition (AGW)

corpus, a collection of over 1 500 000 text documents, we created a testbed using 90 queries and used them to test AIR effectiveness. Our results show that different AIR systems perform almost equally, but that effectiveness is lower overall than reported results using a smaller text collection. We also confirm that root stemming does not aid retrieval effectiveness with such a large collection, and that using stems as index terms is significantly better than using roots or words.

Among the techniques we investigate to improve retrieval is the choice of parameters for the Okapi BM25 similarity measure. Initial values optimised for the TREC-8 English collection have been used by researchers on TREC 2001 and TREC 2002 Arabic collections [El-Khair, 2003; Darwish and Oard, 2003a; Darwish et al., 2005]. We show that these values are not the best choice for the Arabic TREC collections and the Arabic AGW collection. We determine that these values differ across collections and should be determined for every individual collection, and that when using short queries, the b parameter has the most effect on retrieval performance and should be determined.

1.3 How effectively can foreign words be identified in Arabic text?

Another category of words in Arabic text that have different spelling variants are foreign words. Foreign words are words that are borrowed from other languages and transliterated into Arabic as they are pronounced by different Arabic speakers, with some segmental and vowel changes. Such words are increasingly common due to the inflow of information from foreign sources, and include terms that are either new and have yet to be translated into native equivalents, or proper nouns that have had their phonemes replaced by Arabic ones. This process often results in different Arabic spellings for the same word. Current AIR systems do not handle the problem of retrieving the different versions of the same foreign word [Abdelali et al., 2004], and instead typically retrieve only the documents containing the same spelling of the word as used in the query. Stemming is not beneficial with such words, as they have no clear affixes. In fact, stemming would be harmful, since core letters that match Arabic affixes would be removed, resulting in the word being mapped to another index term. Before dealing with such variants in Arabic, an essential first step is to identify them. We manually extract a list of foreign words from a large collection of Arabic text, and evaluate three techniques to identify these: lexicons, patterns and n-grams. We enhance the lexicon-based technique using rules based on the structure of Arabic words and letter frequency in both Arabic and foreign words. We also improve the n-gram technique originally

used in language-identification applications, and use it along with the lexicon approach to identify 80% of all the foreign words with a precision of 93%.

1.4 What is the effect of normalising foreign word variants?

Techniques other than stemming are required to group variants of a foreign word under one index term. Normalising different variants under one encoding form, and computing similarity based on n-grams, are often used to find different versions of names in English [Zobel and Dart, 1995; 1996; Christen, 2006a].

Finding variants of names in languages such as English is a problem that has been long recognised in information retrieval, and has been addressed in great depth by the database community [Raghavan and Allan, 2005]. Most experiments have been carried out using name databases [Zobel and Dart, 1995; 1996; Pfeifer et al., 1995; 1996; Pirkola et al., 2002; Holmes and McCabe, 2002; Holmes et al., 2004; Ruibin and Yun, 2005; Christen, 2006a;b]. Results reported using such databases are not reliable for finding name variants within a text environment, as other words found in the text would affect results. For example, words such as "better" and "patter" would be considered similar to the proper noun "Peter" by some phonetic-matching algorithms such as the Soundex or Phonix. Few studies have tested the retrieval of name variants in the context of IR where names are to be located within text documents rather than from a list of names [Raghavan and Allan, 2004; 2005]; moreover, there is only one study that tests the effects of finding Arabic name variants within a list of modified Arabic names [Aqeel et al., 2006]. We test the effects of using string and phonetic similarity techniques to find variants of foreign words in an IR context.

We evaluate the major approaches and introduce others with the aim of identifying variants of foreign words in two collections of Arabic text. We also test the effectiveness of converting variants to a single normalised form. We show that normalising foreign word variants using our algorithms increases recall significantly by 5.04% and increases precision by 9.64% but not significantly. Using query expansion, we show that our phonetic Soutex4 algorithm is the best candidate to expand queries with foreign word variants, with significantly improved precision and recall.

1.5 Thesis Overview

We organise our thesis as follows:

In Chapter 2, we present an overview of the Arabic language, describe the fundamental elements of information retrieval (IR) research.

In Chapter 3, we review prior work on Arabic information retrieval systems, focusing on morphological analysers, light stemmers, and statistical approaches. We also review potential approaches that can be used to match foreign word variants, and those that can be used to distinguish foreign words from native words in the text.

In Chapter 4, we focus on the effects of stemming on AIR. We compare existing AIR systems and propose techniques that avoid stemming core letters in Arabic words. We also investigate the use of language morphological rules to improve stemming. We demonstrate that our rules are more effective using the list of unique words in the collection. We investigate whether the effectiveness of applying techniques used to improve Arabic retrieval using clean text documents applies to text documents generated automatically from an audio soundtrack and using queries translated from English.

In Chapter 5, we build a new test collection using a document collection that contains over 1 500 000 documents. We build 90 queries and draw up associated relevance judgments, and use this collection in testing the effectiveness of AIR systems. We determine the best parameters of the Okapi BM25 function that lead to the highest results with TREC 2001 and 2002 collections as well as our new larger AGW collection.

In Chapter 6, we explore approaches to identify foreign words in Arabic text. We test using lexicons, Arabic patterns and n-grams to distinguish foreign words from Arabic ones. We show that a combined n-grams and lexicon technique is highly effective for this purpose.

In Chapter 7, we test the effects of string- and phonetic-similarity techniques in finding foreign word variants in Arabic text. We empirically show that normalising such words in Arabic text increases precision and recall.

We conclude the thesis in Chapter 8 with a review of the contributions of our research, and a discussion of future research directions.

Chapter 2

Background

The main objective of an Information Retrieval (IR) system is to retrieve documents most relevant to the user's query, and the best IR system best ranks the more relevant documents above less relevant ones. Documents are usually ranked based on terms in the query and terms in the retrieved documents. In many cases, queries do not contain enough terms to disambiguate the user's information need, so the IR system may return irrelevant documents. Many of the techniques developed to improve IR systems retrieval effectiveness in other languages can also be applied for Arabic Information Retrieval (AIR) systems; however, techniques specifically tailored for Arabic are also required. In this chapter, we introduce the Arabic language and explain its structure, and review techniques applied to improve both IR systems in general and AIR systems in particular.

2.1 The Arabic Language

Arabic is the official language of 23 countries, and one of the official languages of the United Nations. It is estimated that with approximately 422 million native speakers, Arabic is the most widely spoken language after Chinese.[1] Arabic is a Semitic language, and a descendant of Proto-Semitic [Bishop, 1998]. The language record goes back to the fourth century BC [Ostler, 2005]. It was developed in the Arabian peninsula and spread out in the seventh century when Islam spread to Asia, Africa and Europe [Jiyad, 2005].

Classical Arabic "فُصحى" /fusˤħa/ — is also formally called modern standard Arabic (MSA) — is the formal language in the Arabic world for reading and writing, and is viewed as the only true version of the language by all Arabs [DeYoung, 1999]. MSA is used to

[1] http://encarta.msn.com/encyclopedia_761570647_4/Language.html

write all books, newspapers, magazines, and media text. However, MSA is not spoken in any country; rather colloquial languages with different dialects are used in each country. Each dialect has its own new terms such as those borrowed from other languages [Bishop, 1998].

2.1.1 Character sets

Arabic is written from right to left in a cursive, consonantal script that has 28 characters. Arabic characters change shape based on their position within words. This extends the Arabic alphabet to ninety different character representations [Tayli and Al-Salamah, 1990]. An Arabic letter might have four different shapes: isolated, initial, medial, and final. In computer encoding systems, the different representations of a character are often mapped to a single base code. for example, the letters "ـم", "مـ", "ـمـ", and "م" are four different shapes of the same letter /mijm/. The computer user does not have to think about these codes as they are generally represented by one code — although different shapes — "E3" in the CP1256 windows coding and "U+0645" in the UTF8 coding.[2] Table 2.1 shows the Arabic alphabet along with their international phonetic association (IPA) symbols that we use to represent the pronunciation of Arabic words [IPA, 1999]. Diacritics are used to clarify the pronunciation of characters within an Arabic word; some can appear with any characters, while others appear only with a limited subset. For example the diacritic hamza "ء" /ʔ/ is used by itself and is also used with the letters "أ", "و", and "ى". Three diacritics are used to represent short vowels that can be used with every consonant character. They mark the consonant to clarify its pronunciation. For example, the consonant "ف" /f/ with the diacritic *fatha* "فَ", is pronounced /fa/, with the diacritic *damma* "فُ", is pronounced /fu/, and with the diacritic *kasra* "فِ" is pronounced /fi/. Two identical diacritics when placed above or below the last letter of an Arabic noun indicate the sound /n/; this is called *tanween*. For example the word "قِصَةٌ" ⟨story⟩ is pronounced /qisʔaṭun/, "قِصَةً" is pronounced /qisʔaṭan/, and "قِصَةٍ" is pronounced /qisʔaṭin/. The diacritic *shadda* as in "فّ" /ff/ is used to marks the gemination (doubling) of a consonant. For example, in the word "رَدّ" (/raḍḍ/⟨returned⟩), the diacritic *shadda* indicates that the letter "د" is found twice in this word and should be stressed. The diacritic *sukoon* is a small circle that is placed above the letter, as in "فْ", indicating a vowel-less consonant. It is used to close an Arabic syllable by marking the closing consonant. This is usually used in unvocalised text to clarify ambiguity of pronouncing Arabic words. For example the words "دَرَسَ" /ḍarasa/ means ⟨studied⟩, but "دَرْس" /ḍars/ means ⟨a

[2] http://www.microsoft.com/globaldev/reference/sbcs/1256.mspx

I	F	M	L	IPA	I	F	M	L	IPA	I	F	M	L	IPA
ء	–	–	–	/ʔ/	ر	ر	ـر	ـر	/r/	ف	ف	ـفـ	ـف	/f/
ا	ا	ـا	ـا	/aa/	ز	ز	ـز	ـز	/z/	ق	ق	ـقـ	ـق	/q/
ب	بـ	ـبـ	ـب	/b/	س	سـ	ـسـ	ـس	/s/	ك	كـ	ـكـ	ـك	/k/
ت	تـ	ـتـ	ـت	/t̪/	ش	شـ	ـشـ	ـش	/ʃ/	ل	لـ	ـلـ	ـل	/l/
ث	ثـ	ـثـ	ـث	/θ/	ص	صـ	ـصـ	ـص	/sˤ/	م	مـ	ـمـ	ـم	/m/
ج	جـ	ـجـ	ـج	/ʒ/	ض	ضـ	ـضـ	ـض	/d̪ˤ/	ن	نـ	ـنـ	ـن	/n/
ح	حـ	ـحـ	ـح	/ħ/	ط	طـ	ـطـ	ـط	/t̪ˤ/	ه	هـ	ـهـ	ـه	/h/
خ	خـ	ـخـ	ـخ	/x/	ظ	ظـ	ـظـ	ـظ	/ðˤ/	و	و	ـو	ـو	/w/
د	د	ـد	ـد	/d̪/	ع	عـ	ـعـ	ـع	/ʕ/	ي	يـ	ـيـ	ـي	/j/
ذ	ذ	ـذ	ـذ	/ð/	غ	غـ	ـغـ	ـغ	/ɣ/	ى	–	–	ـى	/aa/
ة	–	–	ـة	/t̪/	–	–	–	–		–	–	–	–	–

Table 2.1: Different shapes of Arabic letters when they come isolated, "I"; as a first letter, "F"; in the middle of the word, "M"; or as a last letter, "L". The IPA column shows their international phonetic representation. The first letter "ء" can come with other characters such as "ا", "و" and "ي". The letter ة *can also be pronounced as the letter* ه*.*

lesson⟩. Without the diacritics, a reader might mistake the two forms.

In general, diacritics are not indicated; readers must rely on context to determine implicit diacritics, and how the word should be pronounced. For example, some of the variants of the word "كتب" are "كَتَبَ" (/kataba/⟨he wrote⟩), "كُتُب" (/kutub/⟨books⟩), or "كُتِبَ" (/kutiba/⟨is written⟩).

The *tatweel* (also known as *kashida*), "ـ", is a special character that is commonly used in typeset Arabic text. This character is not an actual letter, as it is used only for cosmetic purposes [Goweder and Roeck, 2001]. It can be inserted between any two concatenating letters. For example, the word "قال" (/qaala/⟨said⟩) can be written as "قـال", "قـــال", and even "قـــــــال". Notice that in this word the *kashida* can only come between the letters "قـ" and "ـا", as they are the only two letters that change shape when connected to each other. Letters that do not change shape when connected to other letters are "ا", " "د", "ذ", "ر", "ز", "و", and "ى".

Unlike English, Arabic has no capital letters, and most proper nouns contain no orthographic signs to distinguish them from other words.

2.1.2 Grammar

In this section we introduce concepts of Arabic grammar that are not found in English and that may have an impact on information retrieval.

Arabic words are categorised into three categories: nouns, verbs and particles. A noun is a word that has a meaning without any association with time. Nouns are either definite or indefinite. Definite nouns are proper nouns; nouns preceded by the definite article "ال" (/al/⟨the⟩); personal pronouns such as "أَنَا" (/ʔanaa/⟨I⟩), and "أَنتَ" (/ʔanṯa/⟨you⟩); demonstrative pronouns such as "هَذَا" (/haḏaa/⟨this⟩), and "هَذِهِ" (/haḏihi/⟨this -feminine-⟩); relative pronouns such as "الذِي" (/alḏij/⟨which -masculine-⟩), and "التِي" (/alṯij/⟨which -feminine-⟩⟩); and the genitive construct, where one noun is determined by another, as in "كِتَابُ المُعَلِّمِ" (/kiṯaabu lmuʕllmi/⟨the book of the teacher⟩), where the noun "كِتَاب" (/kiṯaab/⟨a book⟩) is made definite by its relationship to the definite noun "المُعلِّمُ" (/ʔlmuʕllimu/⟨the teacher⟩).

A verb is a word that indicates an action at a certain time. Verbs are either perfect or imperfect (present or future tense). Perfect verbs denote completed events, while imperfect verbs denote uncompleted actions. Verbs are inflected and morphologically marked according to person, number, and voice (active and passive). Imperfect verbs are also inflected according to mood (indicative, subjective, jussive, and imperative) [Yagoub, 1988].

Any word that is not a noun or a verb is categorised as a particle. Particles are words that have no meaning by themselves, for example prepositions and conjunctions.

Arabic has two types of sentences: nominal, and verbal. A nominal sentence is a sentence that starts with a noun, while a verbal sentence is a sentence that starts with a verb. In both sentences there should be an agreement in number and gender between the verb and the subject.

Arabic has two genders, usually referred to as masculine and feminine. The suffix marker for the feminine gender is a "ة". The feminine form of the word is usually formed by adding this suffixes to the masculine singular form. For example, "مُدَرّس" (/muḍarris/⟨a teacher⟩) is a masculine singular word, "مُدَرّسة" (/muḍarrisah/⟨a teacher⟩) is the female form. Although most feminine forms are formed the same way, there are exceptions where feminine words do not actually end with the feminine suffix as in "الشَمس" (/aʃʃams/⟨the sun⟩) and "الصَحراء" (/alsˤaħraʔ/⟨the desert⟩).

Arabic has singular, dual, and plural forms, each with its own pronouns and suffixes. The dual form is usually formed by adding the suffix "ان" to the singular form. For exam-

ple, "مُدَرّس" (/muḍarris/⟨a teacher⟩) is a singular form from which the dual form "مُدَرّسان" (/muḍarrisaan/⟨two teachers⟩) is generated by adding the dual suffix. There are two types of plurals in Arabic: regular plurals — known as sound plurals — and irregular or "broken" plurals. The regular plural is formed by adding a specific suffix to the singular form of the noun. The masculine sound plural is formed by adding the suffix "ـون" to the singular form, while the feminine sound plural is formed by replacing the singular feminine suffix "ـة" with "ـات". For example, "مُدَرّسون" (/muḍarriswn/⟨teachers⟩), and "مُدَرّسات" (/muḍarrisaaṭ/⟨teachers -feminine-⟩) are the masculine sound plural and the feminine sound plurals for the singular "مُدَرّس" (/muḍarris/⟨a teacher⟩) respectively. Irregular plural are formed using patterns rather than the regular suffixes. For example, the word "اعلام" (/ʔʕlaam/⟨flags⟩) is the plural form of the word "عَلَم" (/ʕalam/⟨a flag⟩), the word "كُتُب" (/kuṭub/⟨books⟩) is the plural form of the word "كِتَاب" (/kiṭaab/⟨a book⟩), and the word "رُسُل" (/rusul/⟨messengers⟩) is the plural form of the word "رَسول" (/raswl/⟨a messenger⟩).

There are different pronouns to address each gender and number for the first, the second and the third person. We discuss pronouns further in the following section.

2.1.3 Morphology

In this section we introduce the morphology of the Arabic language, and lay the groundwork for our discussion of techniques to improve the effectiveness of AIR systems.

Arabic has a rich morphology that cannot be fully described in one chapter. We only describe issues related to the word structure that we can apply in removing affixes and returning words to their root or stem. For a detailed treatment of Arabic grammar, we recommend the works of Yagoub [1988] and Wright [1874].

As in other Semitic languages, Arabic words are formed by applying vowel patterns to roots that have three or four — and in rare cases five — letters. Roots are the basic form of Arabic words. They cannot be derived from any Arabic words and usually describe the basic lexical meaning of the word. There are 6,350 triliteral roots and 2,500 quadrilateral ones listed in "لسان العرب" /lisaanu lʕrab/, one of the most respected Arabic dictionaries [Moukdad, 2006], but Beesley [1996] reported that there are around 5,000 roots used in modern standard Arabic.

Stems are roots combined with derivational morphemes — generally using patterns — that attach to a word at the beginning (prefix), the middle (infix), or the end (suffix). Stems are the basic form of a surface word that can be inflected using other morphemes.

For example, the word "دُرُوس" (/ḍuruws/⟨lessons⟩) is a stem comprises the root"دَرَسَ" (/ḍarasa/⟨studied⟩) and the infix "و".

Surface forms of Arabic words comprise two or more morphemes: a root with a semantic meaning, and a pattern with syntactic information [Aljlayl, 2002]. There are around 400 distinct patterns in Arabic [Beesley, 1996]. The most well-known pattern is "فعل" (/faʕla/⟨he did⟩), which is often used to generically represent three-letter root words. For example: the root "كتب" (/kaṯaba/⟨wrote⟩) can be represented by the pattern "فعل" by mapping "ك" to "ف", "ت" to "ع", and "ب" to "ل". Characters are added at the beginning, the middle, or end of the root, but the base characters that match the pattern remain unchanged. For instance, "فعال", "فاعل", and "يفعل" are three patterns to respectively form the singular noun, the active participle, and the present tense verb out of the pattern "فعل". By fixing the core letters and adding additional letters to each pattern, we can generate "كتاب" (/kiṯab/⟨a book⟩), "كاتب" (/kaṯib/⟨writer⟩), "يكتب" "يكتُب" (/jkṯub/⟨he writes⟩) respectively. Note that all derived forms are related to the concept of writing contained in the root word. Similarly, many words can be formed from the root "صَنَعَ" (/sˤanaʕa/⟨he made⟩) that relate to the concept of making; for example, "صِناعه" (/sˤinaʕh/⟨Manufacturing⟩) , "صانِع" (/sˤaniʕ/⟨a handcraft man⟩), and "يصنَع" (/jsˤnaʕ/⟨he makes⟩).

A lemma is similar to the root. It represents a set of surface forms that share the same meaning. However, the root is broader in that it might also represent words with different meaning. For example, the word "فَجر" (/faʒr/⟨Dawn⟩) and "انفِجار" (/infiʒaar/⟨explosion⟩) share the same root "فجر" /fʒr/ [Kamir et al., 2002]. In fact, in the absence of diacritics, It is hard to differentiate between the lemma and the root in Arabic.

Nouns are inflected and morphologically marked according to gender (masculine or feminine); case (nominative, genitive, or accusative); number (singular, dual, or plural); and determination (definite or indefinite) [Yagoub, 1988]. An example of inflecting the noun "مُعَلِّم" (/muʕallim/⟨a teacher⟩) is shown in Table 2.2. Foreign words are nouns that do not follow these inflection rules.

Arabic words accept prefixes and suffixes. In contrast to English, most connectors, conjunctions, prepositions, pronouns, and possessive pronouns are attached directly to the Arabic word, forming more complicated derivations. Infixes are added to nouns by applying patterns, often to form broken plurals. A combination of these affixes results in many different forms for the same word. For instance, Chen and Gey [2002] presented 86 different forms for the word "طِفل" (/tˤifl/⟨a child⟩), and more can be formed. Attia [2005] generated 1,800

	Masculine			Feminine		
	Nominative	Genitive	Accusative	Nominative	Genitive	Accusative
Singular	معلمٌ	معلمٍ	معلمًا	معلمةٌ	معلمةٍ	معلمةً
Dual	معلمان	معلمَين	معلمَين	معلمتان	معلمتين	معلمتين
Plural	معلمون	معلمِين	معلمِين	معلماتٌ	معلماتٍ	معلماتٍ

Table 2.2: Inflected forms of the noun "مُعَلِّم": all words accept the definite article for determination, other prefixes such as prepositions and conjunctions, and suffixes such as possessive pronouns. In the absence of diacritics, only 9 unique forms remain.

sound versions of the verb "شَكَرَ" (/ʃakara/⟨to thank⟩) and 519 sound versions of the noun "معلم"(/muʕallim/⟨a teacher⟩).

Particles can also accept affixes. They form a clitic when they are expanded with affixes [Attia, 2007]. For example, "له" ⟨his⟩ is a clitic composed of the preposition "لـ" and the personal pronoun "ـه". Some particles can appear on their own, while others — known as inseparable particles — can only be used attached to other words. Prepositions are an important type of particle; there are twenty prepositions in Arabic, five of which are inseparable. These are "و", "ك", "ل", "ب", and "ت".

2.1.4 Arabic Affixes

As presented in the previous section, all Arabic words are generated from root words. This is usually done by adding vowels to the root words to form the stem. The stem is inflected by adding prefixes, infixes and suffixes. As Arabic is written from right to left, prefixes are added to words from the right side and suffixes are added at the left side. For example, the word "والطالِبان" (/walṭˤaalibaan/⟨and the two students⟩) has the prefix "والـ" on the right and the suffix "ـان" on the left. Generally, ten letters are used in Arabic affixes: "س", "أ", "ل", "ت", "و", "م", "ن", "ي", and "ه"; these are grouped in the acronym "سالتمونيها". Some prefixes and suffixes may be used in combination with both nouns or suffixes, while others are used exclusively with nouns or with verbs. We follow with a discussion of these three types of affixes.

Common Affixes:

Common prefixes and suffixes can attach to nouns and verbs. In some cases, they can also attach to some particles. We present these affixes and present exceptions where appropriate.

Common Prefixes: Conjunctions are the only type of common prefix in Arabic, and can be added to any word. The most frequent conjunctions are "و" ⟨waw⟩ and "فـ" ⟨faa⟩. These two conjunctions attach to any word directly. There are many words that contain these characters in their core (not as affixes); for example the word "وَفِي" (/wafij/⟨sincere⟩) starts with "و" as a core letter. In systems where surface words are usually extracted, this creates ambiguity. If the first letter is removed the word becomes "فِي"(/fij/⟨in⟩), which is a preposition. Such ambiguity occurs frequently in Arabic. The letter lam "لـ" can be used for different types of particles. In addition to its purpose as a preposition, which makes it a noun prefix, it can also be used with verbs as the "lam of command". Here, it is usually prefixed to the third person to give it an imperative sense, for example لِتَقُلهَا (/liṯaqulhaa/⟨say it⟩). It is also used to indicate the purpose for which an action is performed [Wright, 1874]. As a particle, it can also be combined with pronouns to form a clitic. This prefix is even more frequent than the conjunction "فـ" [Chen and Gey, 2002]. An AIR system must handle each type of particle — and letters that falsely appear to be particles — appropriately.

Common Suffixes: First-, second-, and third-person pronouns are common suffixes and can be attached to nouns, verbs, and some particles. Table 2.3 shows how these suffixes are used with the word "قلم". Third-person pronouns are more frequent than the first and the second personal pronouns in written Arabic text, as the last two are mostly used in speech. This is clearly shown by Chen and Gey [2002] in the most frequent one, two, and three suffixes in the TREC 2001 corpus.

Another suffix common to both nouns and verbs is the suffix "ون", which is added to the masculine singular form to indicate the nominative masculine sound plural. For example, "مُعَلِمون" (/muʕalimwn/⟨teachers⟩) is the sound plural of the singular "مُعَلِم" (/muʕalim/⟨a teacher⟩). Masculine sound plurals are similarly formed by adding the suffix "ـين" (Table 2.2).

This suffix also attaches to present tense verbs to indicate the plurality of the sentence subject. For example "يَسمَع" (/jasmaʕ/⟨listens⟩) turns to "يَسمَعون" (/jasmaʕwn/⟨they -masculine- listen⟩). If the present tense verb is used in the jussive mood, this prefix is replaced with "ـوا", which also used when the verb is in the imperative or the past tense.

		1st Person		2nd Person		3rd Person	
		Word	Meaning	Word	Meaning	Word	Meaning
Singular	Mascu.	قلمـ ـي	⟨my pen⟩	قلمـ ـكَ	⟨your pen⟩	قلمـ ـه	⟨his pen⟩
	Femin.	قلمـ ـي	⟨my pen⟩	قلمـ ـكِ	⟨your pen⟩	قلمـ ـها	⟨her pen⟩
Dual	Mascu.	قلمـ ـنا	⟨our pen⟩	قلمـ ـكما	⟨your pen⟩	قلمـ ـهما	⟨their pen⟩
	Femin.	قلمـ ـنا	⟨our pen⟩	قلمـ ـكما	⟨your pen⟩	قلمـ ـهما	⟨their pen⟩
Plural	Mascu.	قلمـ ـنا	⟨our pen⟩	قلمـ ـكم	⟨your pen⟩	قلمـ ـهم	⟨their pen⟩
	Femin.	قلمـ ـنا	⟨our pen⟩	قلمـ ـكن	⟨your pen⟩	قلمـ ـهن	⟨their pen⟩

Table 2.3: Common pronoun suffixes can appear with nouns or verbs; in this example, we show the word "قلمـ" ⟨pen⟩. This word can be replaced by other nouns and verbs. When using verbs, the singular suffix "ـي" under the 1st person should be changed to "ـني" ⟨me - object⟩, and all English possessive adjectives should be replaced with object pronouns.

Noun Affixes:

Nouns can have prefixes, infixes and suffixes. Prefixes and suffixes attach to a noun without changing its structure, while infixes are added irregularly using construction patterns.

Noun Prefixes: The most common noun prefix is the definite article "ال" /al/. This prefix — like the English "the" — comes before nouns only. It can be preceded by conjunctions and prepositions. The frequency of this prefix in Arabic text is very high. Chen and Gey [2002] reported this prefix to be the most frequent initial two- or three-character sequence in the TREC 2001 collection. When this prefix is preceded by the preposition "لـ", they combine to form the prefix "للـ" /ll/. For example, the noun "البَيت" (/albajt̪/⟨the house⟩) becomes "للبَيت" (/llbajt̪/⟨to the house⟩).

Prepositions are another category of prefixes that are specific to nouns only. Separable or isolated prepositions are words written independently, while inseparable ones such as "و", "لـ", "كـ", "بـ", and "تـ" are attached directly to Arabic nouns. As discussed previously, the particle "لـ" can appear with verbs but not as a preposition. Similarly, the particle "و" can be a conjunction that precedes any word. The preposition "تـ" is rarely used in modern Arabic, but appears very commonly as a verb prefix. The remaining inseparable prepositions "بـ" and "كـ" can only be used with nouns. Based on the TREC 2001 corpus statistics, the most frequent particle in Arabic is "و", followed by "بـ", "لـ", and 'كـ" [Chen and Gey, 2002].

Noun Infixes: In addition to the sound masculine and sound feminine plural forms, the broken plural form of Arabic nouns is formed irregularly from singular nouns using patterns. There are no fixed prefixes, or suffixes. Instead, most additional letters are infixes, usually vowels. In some cases, the singular form does not change, and only the diacritics change, causing the plural form to be pronounced differently. In other cases, some letters are removed from the singular form to obtain the plural. Some examples of broken plurals are: "رِجَال" (/riʒaal/⟨men⟩) from the singular "رَجُل" (/raʒul/⟨a man⟩), "أقلَام" (/ʔqlaam/⟨pens⟩) from the singular "قَلَم" (/qalam/⟨a pen⟩), and "غُرَف" (/ɣuraf/⟨rooms⟩) from the singular form "غُرفَة" (/ɣurfaṯ/⟨a room⟩). Broken plurals are generated using patterns, and it is generally possible to return the plural to the singular form by reversing the process. However, there is some ambiguity associated with this process, since the clarifying diacritics are generally absent in Arabic text. Broken plurals constitute about 10% of all words in large Arabic corpora [Goweder et al., 2004].

Noun Suffixes: The dual suffix "ـان" comes only with nouns. This suffix is added to the singular form of the noun to form the dual form. For example, the word "الطَالِب" (/alṭˤaalib/⟨the student⟩) is in the singular form, while the word "الطَالِبَان" (/alṭˤaalibaan/⟨the two students⟩) is the dual form. When changing the feminine singular form to the dual form, the last letter, used to indicate feminity "ـة", is usually changed to "ـتـ" before adding the suffix "ـان". For example, the word "الطَالِبَة" (/alṭˤaalibaṯ/⟨the female student⟩) is in the singular form, while the word "الطَالِبَتَان" (/alṭˤaalibaṯaan/⟨the two female students⟩) is in the dual form. This suffix is usually replaced by the suffix "ـين" if the noun comes in the genitive or the accusative mood.

The feminine suffixes "ـة" and "ـات" are used to represent the feminine singular and the plural respectively. As discussed in the previous paragraph, the feminine sound plural is formed by changing the suffix "ـة" to "ـات". For example, the word "الطَالِبَات" (/alṭˤaalibaaṯ/⟨the female students⟩) is the plural form of the word "الطالبة".

The possessive pronoun "ـي", can also be an attributive pronoun which attaches only to nouns. For example, the word "عَرَبِي" (/ʕarabij/⟨Arabic⟩) is an adjective from the word "عَرَب" (/ʕarab/⟨Arab⟩), and attributes the subject being described, such as a person or language, to the word "Arab".

Verb Affixes

Verbs can have both prefixes and suffixes. Most suffixes such as pronouns are common between verbs and nouns, which we have presented in the common affixes (Section 2.1.4). However, there are some affixes specific to verbs.

Verb Prefixes: These are prefixes that can only appear before verbs, and usually indicate that the word is a present-tense verb. The most common of these prefixes are represented in the acronym "انيت". The prefixes "ا", and "نـ" are used to refer to the first-person singular and plural forms respectively, as in "أَكُلُ" (/ʔakulu/⟨I eat⟩), and "نأَكُلُ" (/nʔakulu/⟨we eat⟩); while the prefix "يـ" is used to refer to the masculine third person as in "يَشرَبُ" (/jaʃrabu/⟨he drinks⟩); and the prefix "تـ" is used to refer to the feminine third person as in "تَشرَبُ" (/taʃrabu/⟨she drinks⟩). To indicate future tense, the prefix "سـ" is added before these prefixes. For example "سَيَشرَبُ" (/sajaʃrabu/⟨he will drink⟩). These prefixes are usually added to the past tense verbs to form the present tense verbs without any changes in the original form. However, in some cases, where the past tense of the verb has the letter "ا", "و", or "ي" (called *weak* letters), the structure of the original verb changes. For example, the present tense verb "يشرب" is a result of combining the prefix "يـ" with the past tense verb "شرب" ⟨drank⟩, but the present tense verb "يَقُولُ" (/jaquwlu/⟨he says⟩) is a result of combining the prefix "يـ" with the past tense verb "قَالَ" (/qaala/⟨he said⟩). Note that the middle letter "ـا" is changed to "ـو" in the present tense form after adding the prefix "يـ".

Verb Suffixes: Some suffixes are used only with verbs, and never with nouns or particles; one of these is "ـت", which is appended to a past-tense verb to refer to the subject (actor) that made the action. This could refer to the first-person as in the word "أَكَلتُ" (/ʔakalṯu/⟨I ate⟩), to the second-person as in "أَكَلتَ" (/ʔakalṯa/⟨you ate⟩), or the third-person feminine as in "أَكَلتْ" (/ʔakalṯ/⟨she ate⟩). The only difference between the last three words is the diacritic over this prefix. In the absence of diacritics, the three look exactly the same. This suffix can also be followed by a third person pronoun as an object, to form a complete sentence. For example, "أَكَلتُهم" (/ʔakalṯuhum/⟨I ate them⟩), "أَكَلتَهم" (/ʔakalṯahum/⟨you ate them⟩), "أَكَلتْهم" (/ʔakalṯhum/⟨she ate them⟩). More complex forms can be formed especially when the "ت" suffix refers to the second person.

Another suffix that appears only with verbs is the second-person feminine pronoun "ي",

as in the word "تَأْكُلِي" (/ṯaʔkulij/⟨you are eating -feminine-⟩); object suffixes can further be added.

The suffix "وا" is used to refer to the masculine plural. This can come with the imperative, past tense, or present tense verbs. It refers to the second-person when it comes after an imperative verb, while it refers to the third-person when it comes after the present or past tense verbs. In the present tense, this suffix replaces the sound plural suffix "ـون" if the mood of the verb changes to jussive.

2.1.5 Foreign Words in Arabic Text

Words are translated between languages, and many words that appear in one language are acquired by another. Translated words are usually modelled to conform to the conventions of the target language. However, some words such as proper nouns and technical terms are not easily or usefully translated, and are instead transliterated into the characters of the target language. To do so, the pronunciation of the original word is converted into the phonemes of the target language through *transliteration*. However, phonetics can differ across languages and not all the phonemes of the source language may exist in the target language [Alghamdi, 2005], so some approximation is often necessary. Transliteration often results in multiple spellings for the same word. This is an issue even across languages that use substantially the same character set; simple examples would be "colour" and "color" across British and American usage, and "ambience" and "ambiance" across French and English. A change in character sets compounds the problem [Alghamdi, 2005; Halpern, 2007; Kashani et al., 2007; Stalls and Knight, 1998]. For instance, Arbabi et al. [1994] reported that the name "سُلَيمان" (/sulajman/⟨Sulayman⟩), which has only one form in Arabic, is written in as many as 40 different forms in English, among them are "Sulyman", "Soliman", and "Sullaiman".

Words translated into Arabic — sometimes referred to as Arabised words [Aljlayl and Frieder, 2002] — are foreign words that are modified or remodelled to conform to Arabic word paradigms, and are well assimilated into the language. The assimilation process includes changes in the structure of the borrowed word, such as segmental and vowel changes, addition or deletion of syllables, and modification of stress patterns [Al-Qinal, 2002]. For example, the words "فيروس" ⟨virus⟩, "ارشيف" ⟨archive⟩, and "راديو" ⟨radio⟩ are originated from other languages, but have a single version in Arabic. Where equivalent native terms are not available early enough for widespread adoption, foreign terms are used directly with their original pronunciation represented using Arabic letters. These do not appear in standard

ملسفتش	/mlsftʃ/	no diacritics; pronunciation unclear
مِلُسُفِتش	/milusufitʃ/	diacritics clarify the correct pronunciation
ميلوسوفتش	/mijluwsuwfitʃ/	long vowels clarify the correct pronunciation

Table 2.4: Diacritics or long vowels used to disambiguate pronunciation for "Milosevic".

Arabic lexicons, and are considered to be Out-Of-Vocabulary (OOV) words.

It should be made clear that not all OOV words are foreign words, nor are all foreign words OOV words. There are many proper nouns that originate from Arabic and follow the Arabic word structure but are not found in Arabic dictionaries. On the other hand, some foreign words have been adopted and are included in Arabic dictionaries. Our main concern in this thesis is foreign words that are characterised by different forms and have no clear standard in writing.

Faced with the need to use new foreign terms, native speakers often cannot wait for formal equivalents to be defined. This is particularly true for news agencies, which encounter new foreign nouns and technical terms daily. This urgency leads to more transliteration than translation, with the associated problem of multiple spellings. In Arabic, short vowels are only indicated using diacritics, but these are rarely used in general text. Context does not help in predicting diacritics for foreign words such as proper nouns or technical terms, and consequently long vowels are often used to make the pronunciation explicit in the spelling of the word without relying on diacritics. This, too, is subject to variation; some transliterators add a long vowel after each consonant in the word, while others add just enough long vowels to clarify word segments with ambiguous pronunciation. Table 2.4 shows how diacritics or long vowels may be used to clarify and specify the pronunciation of the word "Milosevic".

The absence of diacritics in typical written text also creates disambiguation problems in other languages; for example, in Persian, the word "نه" /nh/ can be either نُه (/nuh/⟨the number nine⟩) or "نَه" (/nah/⟨no⟩).

The absence of certain sounds in Arabic, and varying pronunciations across dialects, also contributes to the multiplicity of spellings. Alghamdi [2005] reported that there are 21 phonemes in Arabic that have no equivalent phonemes in English, and the American speech-language-hearing association reported that English phonemes that are not found in Arabic include /p/, /r/, /ʒ/, /g/, and /ŋ/.[3] This causes multiple transliterations for the

[3]http://www.asha.org/nr/rdonlyres/8ac103f3-f7eb-44bd-adb2-afa8aa389327/0/arabicphonemicinventory.pdf accessed on 20th April 2008.

same English phoneme. For example, the phoneme /g/ has no standard equivalent in Arabic; it is at times mapped to the Arabic letters "غـ" /ɣ/, "قـ" /q/, or "جـ" /ʒ/ [Abduljaleel and Larkey, 2003]; we have also observed it mapped to the letter "كـ" /k/: "غورباتشوف", "قورباتشوف", "جورباتشوف", and "كورباتشوف" are among the transliterations of the name "Gorbachev" that we have found on the Web.

Similarly, the interpretation of character combinations varies between transliterators. Moreover, typographical and phonetic errors during transliteration may add even more variants [Borgman and Siegfried, 1992]. The education and the experience of the actual transliterator also contributes to the transliteration result [Arbabi et al., 1994].

2.1.6 Summary

We have introduced the Arabic language and explained its morphology. We have presented the characters used in Arabic and their different representations in Arabic text. We have also explained the different categories of Arabic grammar, and the possible affixes that an Arabic word may take. We classified affixes into three categories: common, noun and verb affixes. Our intention is to return the different forms of an Arabic word to its stem. We will describe how we approach this problem in Chapter 4. We have defined foreign words in Arabic and explained that their structure does not follow any standard, which results in different versions of the same word appearing in Arabic text. We deal with this category of text later in this thesis by presenting algorithms to identify them in Arabic text in Chapter 6, and presenting algorithms to conflate different variants of the same foreign word to one form in Chapter 7. We continue with a review of text retrieval systems in general, and Arabic text retrieval systems in particular.

2.2 Information Retrieval

Information retrieval (IR) is a way to organise, represent and store information items so that the user can access them easily [Baeza-Yates and Ribeiro-Neto, 1999]. Web search engines are a widely used form of information retrieval systems; they collect information by crawling web pages and parsing and indexing their contents. Users typically convey their information need to the search engine in the form of one or more query keywords. The search engine matches these query terms with terms from documents in the collection, and returns documents from the collection in decreasing order of estimated relevance to the query. Information retrieval systems are distinct from data retrieval systems [Zobel et al., 1998; Baeza-Yates

Doc. ID	Document Text
1	The word "القِطّة" means "the female cat"
2	To get rid of the rat, introduce the cat.
3	The sentence "قطٌّ أَرقَط" means "a dotted male cat".
4	We feed the cats to the rats, the rats to the cats, and get the skins for free.

Table 2.5: An example document collection. We use this document collection throughout this chapter to explain different aspects of IR.

and Ribeiro-Neto, 1999]; the latter are used to find data that satisfies clear criteria, while the former estimate likelihood that data is relevant to the query, and rank the data accordingly.

In this thesis, we focus on developing and improving IR techniques for retrieval from collections of Arabic text, although many of the methods we describe are also suitable for data retrieval applications.

In the following subsections, we describe some of the fundamental techniques used by IR systems: parsing, where raw documents are split into proper terms for indexing; indexing, where terms are indexed to facilitate searching; and finally, searching, where user queries are matched against indexed terms and results are ranked. Table 2.5 shows a small sample collection that we use in the next sections to explain internal components of IR systems.

2.2.1 Parsing

Parsing in IR systems involves extracting terms from documents by identifying tokens based on boundary rules, and removing punctuation. During this process many other operations can be applied to the extracted tokens; common operations include spelling correction, normalisation, stopping and stemming.

Term Extraction

Text documents are composed of tokens, separated by spaces or punctuation marks. An IR system must identify and extract these tokens; some tokens may be valid words, while others may be markup, such as HTML tags. In this thesis, we use "word", "term", "token" interchangeably; these are not necessarily valid words in a particular natural language.

Doc. ID	Term Extraction	Normalisation
1	The word القِطّة means the female cat	the word القطه means the female cat
2	to get rid of the rat introduce the cat	to get rid of the rat introduce the cat
3	The sentence قطّ أرقَط means a dotted male cat	the sentence قطّ ارقط means a dotted male cat
4	We feed the cats to the rats the rats to the cats and get the skins for free	we feed the cats to the rats the rats to the cats and get the skins for free

Table 2.6: Effects of term extraction and normalisation on the sample collection shown in Table 2.5. Terms are extracted based on word boundaries — spaces and punctuation. Normalisation — shown in the third column — is performed by changing capital case English letters to lower case, removing diacritics, and replacing the letters "ـة" with "ـه" and "أ" with "ا".

Grune and Jacobs [1994] define parsing as "the process of structuring a linear representation in accordance with a given grammar". The two main parts of this definition are the "linear structure" and the "grammar". To the linguist, the linear structure is the sentence and the grammar can be a set of rules that govern the sentence structure. However, in the IR context, the linear structure could be the document and the grammar could be the rules to split up the text into its component parts. These rules differ between parsers and collections.

Most parsers remove text components that do not contribute to the document content. Such components could be mark-up tags and punctuation. Word and sentence boundaries are determined from punctuation. However, punctuation is language-dependent; for example, question sentences are ended with the symbol "?" in English, but with the symbol "؟" in Arabic. Some languages such as Chinese, Japanese, and Korean (CJK) have no clear word boundaries. These languages are parsed differently using morphemes and n-grams [Vines and Zobel, 1999].

In Arabic, we parse the text based on the sequence of Arabic letters. Spaces and punctuation are used as word boundaries and are usually removed during parsing, as are other characters such as diacritics and the *tatweel*. The second column of Table 2.6 shows an example of extracting terms from the original sample collection presented in Table 2.5.

After token extraction, many operations might be carried out by a parser; these are explained in the following subsections.

Normalisation

Words can be written in different forms; in English, a word may appear capitalised at the start of a sentence, and in lower case elsewhere. For such related words to be associated for retrieval, they must be normalised. In our example, case folding [Witten et al., 1999] can be used to represent the words in a uniform manner.

In Arabic, characters have different shapes, and additional variation is added by differing writing conventions. For example, when the letter "ي" appears at the end of a word, it is usually replaced by the identically-pronounced letter "ى". Another example is the letter "ا", which can be written as "ا", "أ", "إ", or "آ"; many writers write a bare alef, while others write it with the proper diacritic. This causes the same word to look different, and critically, to have a different set of character encodings. For example, "أشرب" (/ʔʃrb/⟨I drink⟩), and "اشرب" (/ʔʃrb/⟨I drink⟩) are the same word, but with a different spelling. Yet another example is the letter "ة", which is sometimes written as "ه". Diacritics are used sparingly in general Arabic text, and so we remove them to unify the vocalised and unvocalised forms. The third column of Table 2.6 shows the effects of normalisation on our sample collection.

Other variations are caused by the lack of writing standards and by differences in dialects; a notable instance of this occurs in the way foreign words are written. We explore this issue in depth in Chapter 7.

Stopping

Words that appear very frequently in a document collection are considered to add little document-specific information. To avoid the noise that is likely to arise from such generic terms, as well as to reduce the size of the index, they are often omitted during the indexing stage [Baeza-Yates and Ribeiro-Neto, 1999]. For example, the articles "a", "an" and "the" in English contribute no information specific to the document topic, as they appear in almost every document in the collection. Removing such words would decrease the index size and improve the search results by leaving words that are more specific to each document. Similarly, the word "فِي" (/fij/⟨in⟩) in Arabic occurs frequently in every document. Generally, particles, pronouns, and function words contribute little information to an Arabic document.

Stopword lists drawn up for Arabic [AlShehri, 2002; Khoja and Garside, 1999] contain well-known pronouns, prepositions and function words. However, these lists differ substantially, and no single widely accepted list exists. Critically, most lists include a single version of each word, despite the fact that Arabic words have different forms. For example, the word

Doc. ID	Stopping	Stemming
1	word القطه means female cat	word قط mean female cat
2	get rid rat introduce cat	get rid rat introduce cat
3	sentence قط ارقط means dotted male cat	sentence قط ارقط mean dotted male cat
4	feed cats rats rats cats get skins free	feed cat rat rat cat get skin free

Table 2.7: Effects of stopping and stemming on the sample collection shown in Table 2.5. The English words "the", "to", "a", "we", "of", "and", "for" are considered stopwords. Stemming is done by removing the plural suffix "s" from English words and the prefix "الـ" and the suffix "ـه" from Arabic words.

"فِي" (/fij/⟨in⟩) is a stopword in almost all Arabic information systems, even though this word occurs in many other forms such as "فِيهِ" (/fijhi/⟨in it -masculine-⟩), "فِيهَا" (/fijhaa/⟨in it -feminine-⟩), "فِيهِمَا" (/fijhumaa/⟨in them -dual-⟩), and so on.

El-Khair [2003] studied this approach and proposed three lists; a general stopword list containing 1,377 words, a corpus-based stoplist with 235 words, and a combination of the previous two with 1,529 words. Chen and Gey [2002] describe a stoplist created by translating 541,681 unique Arabic words to English and then capturing all words that translate to English stopwords. Their list had 3,447 words. Despite this disagreement on the appropriate stopword list size and content, there is an agreement that removing them from Arabic text improves retrieval precision.

Stopwords have to be chosen carefully as they affect retrieval. In English for example, some queries might contain only stopwords, for instance, "to be or not to be". In Arabic, some function words can be spelt identically to proper nouns. The absence of diacritics makes it difficult to distinguish between such words unless we consider the context. For example, the word "على" could be (/ʕalaa/⟨above⟩), and it could be the proper noun (/ʕalij/⟨Ali⟩), the word "منى" could be (/mnni/⟨from me⟩) and it could be the proper noun (/muna/⟨Muna⟩), and the word "فِيهِ" (/fijhi/⟨in him⟩) could be a preposition attached to the third-person pronoun "ـه", and it could be ⟨his mouth⟩, although they are identical in pronunciation and writing.

In Chapter 4, we test how removing automatically expanded versions of stopwords can affect retrieval effectiveness for Arabic text collections.

Stemming

Stemming algorithms are used in information retrieval to reduce different variants of the same word with different endings to a common stem [Paice, 1996]. Stemmers can help information retrieval systems by unifying vocabulary, reducing term variants, reducing storage space, and increasing the likelihood of matching documents [Salton, 1989].

Table lookup and affix removal are two different types of stemming [Frakes and Baeza-Yates, 1992]. In the table lookup approach, words and their stems are stored in a table; each word with an entry in the table is replaced by its corresponding stem. This approach is fast, as it does not require word analysis, but it requires space and some overhead in preparing the table. In contrast, affix removal uses morphological rules to strip off suffixes. Some English stemming algorithms such as the S stemmer, strip off only the suffix "s" to conflate plural and singular forms, and others, such as the one described by Lovins [1986], removes the longest possible suffix, leaving at least two or more characters in the stem. Rather than remove only the longest possible suffix or the plural "s", Porter [1980] identifies and removes multiple suffixes. Table 2.7 shows the effects of both stopping and stemming on our sample collections.

The effectiveness of stemming on English information retrieval has been evaluated in several studies. In an IR experiment, Harman [1991] evaluated the S, Porter and Lovins stemmers using three text collections: the Cranfield collection of 225 queries and 1,400 documents, the Medlars collection of 30 queries and 1,033 documents, and the CACM collection of 64 queries and 3,204 documents. She concluded that the three stemmers did not have any significant improvement in precision and recall. Krovetz [1993] enhanced the Porter stemmer by using a machine-readable dictionary. He modified the stemmer to check words against the dictionary before removing suffixes. His experiment showed that stemming increases the effectiveness of English retrieval systems. Hull [1996] compared a lexical-based stemmer with some other English stemmers including the S, Porter and Lovins stemmers and concluded that the S stemmer is not as effective as other stemmers, and that the lexical-based system is not significantly better than other stemmers, but that it could be successful if it were optimised. He also concluded that prefix removal has a negative impact on retrieval effectiveness in terms of precision and recall.

Popovič and Willett [1992] adapted the Porter stemmer to strip suffixes in the Slovene language, which has a more complex morphology than English. Their experiment showed significant improvements in precision. They also made a comparison using the same stemmer

on the English versions of the queries and collection. Results using the English collection showed that stemming has no effects on retrieval performance. They related the success of the same stemmer on Slovene to its complex morphology.

Savoy [1999] tested the effects of stemming on French text retrieval. He found that stemming and stopword removal significantly improve precision; stopping only improves precision when using the Okapi retrieval model, while stemming improves precision in collections that have more shorter documents than longer ones. He also concluded that a simpler stemmer is more suited to the morphology of the French language than a complex one.

Asian [2007] tested the effects of five stemming algorithms on Indonesian text retrieval. Four of these algorithms use a dictionary, while one does not. She showed that the dictionary-based stemming algorithms performed significantly better than the one that did not use a dictionary. She attributed some of the success of the best-performing algorithm to the use of Indonesian morphological rules.

Stemming has been shown to be more effective for Arabic retrieval than for English. Early research in this area was performed using small collections, and it was not until the TREC 2001 Arabic track that a large data set — albeit far smaller than those at hand for English — became available. Several studies on Arabic retrieval have shown that stemming improves retrieval significantly [Aljlayl and Frieder, 2002; Larkey et al., 2002; Chen and Gey, 2002; Darwish and Oard, 2003b; Taghva et al., 2005]. This is an unsurprising result as Arabic is characterised by a high inflection ratio [Goweder and Roeck, 2001].

The exact affixes removed vary between stemmers [Aljlayl and Frieder, 2002; Larkey et al., 2002; Chen and Gey, 2002; Darwish and Oard, 2003b; Khoja and Garside, 1999; Taghva et al., 2005], but most stemmers remove affixes by looking up the beginning and the ending of a word in a pre-prepared list of affixes. Most of the current stemmers apply no rules on removing affixes, except to restrict the length of the remaining stem. We present a review of several Arabic stemmers in Section 3.1.

The above studies on non-Arabic stemming suggest that using lexicons and morphological rules improves retrieval performance. There has been little published research on using comprehensive morphological rules to improve Arabic stemming. We believe that stemming Arabic could be improved using morphological rules. In Chapter 4 we test supporting affix removal in light stemming by both morphological rules and lexicons.

Stemming is not always perfect, and can have undesirable results, such as conflating unrelated words together. It is not a viable means for standardising proper nouns, since there is the risk of incorrect conflation [Paik et al., 1993].

N-gram Tokenisation

Tokenisation — through using n-grams — is the process of parsing text using overlapping windows of a fixed size n. Instead of identifying word boundaries in the text, the whole text is split into overlapping tokens of size n, and then indexed. When a user searches the collections, the query is also tokenised using the same window size, and matched against the index. This technique is language independent and robust against spelling mistakes. Using a window of size three, the sentence "This is a book" is parsed into "Thi", "his", "is ", "s a"," a ", "a b"," bo", "boo", "ook". This technique is particularly useful for languages with indistinguishable word boundaries such as the CJK languages.

The n-grams technique can also be used to compare words to determine similarity. In this case, the beginning and the end of the word might be indicated with an additional character added before and after the original string. For example, the trigrams of the word "Arabic" are "Ara", "rab", "abi", "bic"; and the tailed trigrams for the word "Arabic" are "*Ar", "Ara", "rab", "abi", "bic", "ic*" when using the character "*" to mark the beginning and the end of the word [Pirkola et al., 2002].

The n-grams technique is effective in many applications such as spelling error detection and correction, query expansion, inverted and signature files, dictionary look-up, text compression, and language identification [Robertson and Willett, 1998]. It is also useful in parsing and retrieving documents that have non-textual content, such as images [Rickman and Rosin, 1996], text images [Harding et al., 1997], and music [Doraisamy and Rüger, 2003].

We use n-grams in Chapter 4 to retrieve transcribed Arabic text, in Chapter 6 to identify foreign words in Arabic and in Chapter 7 to match foreign words variants.

2.2.2 Indexing

The result of the parsing stage is a list of terms that represent documents in the collection. In order to facilitate searching these terms in an efficient way, an index is created. The index of a book lists the important terms that appear in the book, and the locations where they appear. In information retrieval, we similarly create an index for a collection of documents by identifying the documents that contain key terms that a user might query for. It is possible to index every term in the collection and even rebuild an approximate collection using that index, if we keep locations along with every term [Witten et al., 1999]. This might be useful, but it is costly in terms of space required by the index.

Many techniques are used to compress the index. Stopping and stemming reduce the

Term	(Doc ID,Term Frequency)
cat	<(1,1),(2,1),(3,1),(4,2)>
dotted	<(3,1)>
feed	<(4,1)>
female	<(1,1)>
free	<(3,1)>
get	<(2,1),(4,1)>
introduce	<(2,1)>
male	<(3,1)>
mean	<(1,1),(3,1)>
rat	<(2,1),(4,2)>
rid	<(2,1)>
sentence	<(3,1)>
skin	<(4,1)>
word	<(1,1)>
قط	<(1,1),(3,1)>
ارقط	<(3,1)>

Table 2.8: An example of an inverted list for the stemmed document collection shown in Table 2.7.

number of terms used in the index, and thus reduce the index size. According to Zobel and Moffat [2006], the most efficient index structure for general-purpose querying is the *inverted file* index. In this structure, every distinct term in the collection has a list containing the identifiers of documents that contain the term. An inverted index for the stemmed and stopped collection of Table 2.5 is shown in Table 2.8.

The index contains all the terms in the collection — in our case the stopped and stemmed collection — and is ordered alphabetically. Each term addresses a list of pairs that include the document identifier in which the term is found, and the frequency of the term in that document.

Another indexing option is the use of *signature files*. Each document is allocated a *signature* or a *descriptor* — usually a number of bits that represents the content of the document [Witten et al., 1999]. This is usually generated by hashing every term in the document several times and setting the bits corresponding to the hashing values to one.

When a user enters a query, a signature is generated by hashing the terms in the query, and comparing the result with the document signatures in the index. When a potential match is found, terms in the query are checked against the potential document to confirm that these terms exist, as bits might be falsely set by other terms in the document. Zobel et al. [1998] found that inverted files are superior to signature files in terms of speed, space, and functionality.

We use the inverted file index in our retrieval experiments in the following chapters.

2.2.3 Searching

The main objective of any IR system is to retrieve the right documents for any specific query. While retrieving the exact documents that meet the user needs is difficult, IR systems estimate the likelihood that a document is relevant to the query, and rank the documents in the collection by decreasing likelihood of relevance. To do this, similarity measures are used to compare the query with documents in the collections. There are two common types of query evaluation: Boolean and ranked query evaluation. In the following subsections, we review these retrieval models.

Boolean Queries

Boolean query evaluation uses logical operators to combine terms in the user query [Witten et al., 1999]. The operators "AND", "OR", and "NOT" are combined with the query terms to form a Boolean expression. The relevance of a document to the query is determined using the Boolean expression formed by the query terms and the logical operators. For example, if a user types the query "rats AND cats", the IR system will retrieve all documents in the collections that contain both words. Documents that only contain one of the words without the other will not be retrieved. Using the index shown in Table 2.8, and assuming that the IR system will normalise and stem the query, only document numbers 2 and 4 will be returned, as they are the only documents that contain both the words "rat" and "cat". If the query is "cats OR rats", the same system should retrieve all the documents as they all contain the word "cat". If the user is interested in documents that contain the word "rats" but not documents that contain the word "cats" then the Boolean query should be "rats AND NOT cats". If no logical operators are used, an implicit conjunction (AND) is typically assumed. Boolean querying uses a binary term weighting, which means that the weights are either "0" (not found in the document) or "1" (found in the document).

Untrained users, especially those from non-English-speaking backgrounds, are rarely aware of the Boolean logic used in some search engines. Salton [1998] states that Boolean logic remains inaccessible to many untrained users, and Spink et al. [2001] reported that less than 5% of internet users use logical operators. Chowdhury [2004] notes that the results of the Boolean queries depend on how well users form their queries, with a high probability that the results will be too general or too narrow. Furthermore, a small variation in the query can lead to very different results [Witten et al., 1999].

Ranked Queries

Ranked queries are more natural than Boolean queries. The user does not have to worry about the complex logical structures as in the Boolean queries. Instead, all documents that contain any of the query terms are retrieved, but ranked according to similarity criteria between the terms in the query and the terms in each document. Documents with more matching terms are usually ranked higher than those with fewer matching terms [Witten et al., 1999]. Users can specify which words are not desired in the query, whereby documents with the specified unwanted terms will be discounted. Documents with very low ranking can be removed from the retrieved documents by setting a threshold [Baeza-Yates and Ribeiro-Neto, 1999]. Ranked querying uses a non-binary term weighting; these weights are used by the similarity measure to determine the overall relevance between the document and the user query. IR systems assign weights to query terms by considering two factors: *term frequency* in the document ($f_{d,t}$), and *document frequency* or number of documents in the collection that contain the term (f_t).

Term frequency favours longer documents as they naturally contain more terms than shorter documents. This can be normalised by dividing the term frequency by the document length [Zobel et al., 1998]. Document frequency is useful in limiting the search to only documents that contain terms in the query. According to Zobel and Moffat [2006], the weight of a term t in a document d and a query q can be calculated as:

$$w_{d,t} = 1 + \ln f_{d,t} \tag{2.1}$$

and

$$w_{q,t} = \ln\left(1 + \frac{N}{f_t}\right) \tag{2.2}$$

where N is the number of documents in the collection.

Vector Space Model

First introduced by Salton and Lesk [1968], this model measures the similarity between the query and the documents in the collection by considering the distinct query terms and the distinct terms in each document to occupy n-dimensional vectors, where n is the number of unique terms in the collection. The query vector contains the weights of the distinct terms in the query, and every document vector contains weights of distinct terms in that document. The similarity between two vectors can be simply measured using the dot product. For example, given the query vector $q =< w_{q,1}, w_{q,1}, w_{q,1}, \ldots, w_{q,n} >$ and the document vector $d =< w_{d,1}, w_{d,1}, w_{d,1}, \ldots, w_{d,n} >$, the similarity between the document and the query ($S_{q,d}$) can be computed using the dot product as:

$$S_{q,d} = q \bullet d = \sum_{t=1}^{n} w_{q,t} \times \ w_{d,t} \tag{2.3}$$

where $w_{q,t}$ is the weight of a term t in the query q, and $w_{d,t}$ is the weight of term t in the document d. As described earlier, we avoid bias towards longer documents by dividing the dot product by the Euclidean length of the query vector $|q|$ and the document vector $|d|$, which defines the cosine angle between the query and the document vectors. This measure is called the *cosine* similarity measure.

$$S_{q,d} = \frac{q \bullet d}{|q||d|} = \frac{\sum_{t=1}^{n} w_{q,t} \times w_{d,t}}{\sqrt{\sum_{t=1}^{n} w_{q,t}^2 \times \sum_{t=1}^{n} w_{d,t}^2}} \tag{2.4}$$

The cosine of an angle determines the similarity between the query and the document vector. If they are completely aligned, then the angle is zero, and thus, the similarity is one; conversely, if the angle is 90 degrees, then the query and the document are completely unrelated (at least from the perspective of the query terms). Values in between give the degree of similarity between the two vectors. These values are used to provide the user with a ranked list of results.

Probabilistic Model

The probabilistic model attempts to estimate the likelihood that a given document is relevant to the user's query, and rank the collection documents by decreasing likelihood of relevance [Robertson and Jones, 1976].

	Relevant Documents (R)	Non-relevant Documents ($\acute{R}$)	Total
Term t present (t)	r_t	$f_t - r_t$	f_t
Term t absent($\acute{t}$)	$R - r$	$N - f_t - (R - r_t)$	$N - f_t$
Total	R	$N - R$	N

Table 2.9: Distribution of term t over the relevant and non-relevant documents in the collection. N represents the number of documents in the collection, r_t represents the number of relevant documents containing term t, f_t represents all documents containing t, and R is the total number of relevant documents.

Consider Table 2.9; the conditional probability that a document R is relevant if it contains a term t is given by

$$P(R|t) = \frac{r_t}{f_t}$$

and the probability that a document R is not relevant if it contains term t is given by

$$P(\acute{R}|t) = \frac{f_t - r_t}{f_t}$$

Similarly, the probability that a term t is present in a relevant document is given by

$$P(t|R) = \frac{r_t}{R}$$

and the probability that a term t is present in a non-relevant document is given by

$$P(t|\acute{R}) = \frac{f_t - r_t}{N - R}$$

Using Bayes' theorem, the weight of term t, w_t can be calculated as:

$$w_t = \frac{r_t/(R - r_t)}{(f_t - r_t)/(N - f_t - (R - r_t))} \tag{2.5}$$

Having calculated the term weight and assuming that terms are independent of each other, the weight for a document d is calculated by the product of its term weights

$$w_d = \prod_{t \in d} w_t$$

The main objective is to order documents by estimated relevance according to their weights, not the specific result of the above equation. Therefore, it is often possible to simply express this as a sum of logarithms [Witten et al., 1999]:

$$\sum_{t \in d} \log w_t$$

The main problem with this model is its dependency on relevance judgements. An enhancement to this model has been proposed by Sparck Jones et al. [2000] that does not need pre-judged documents. Their Okapi BM25 measure considers the document frequency (f_t), the number of the documents in the collection (N), the frequency of a term in the document ($f_{d,t}$) and it normalises document length. The equation used to compute the similarity between a document d and a query q is:

$$BM25(d,q) = \sum_{t \in q} \log\left(\frac{N - f_t + 0.5}{f_t + 0.5}\right) \times \sum_{t \in q} \frac{(k_1+1) f_{d,t}}{k_1 \times \left(((1-b) + b \times \frac{|d|}{(avgdl)}\right) + f_{d,t}} \times \frac{(k_3+1) f_{q,t}}{k_3 + f_{q,t}} \tag{2.6}$$

where $|d|$ is the document length, $avgdl$ is the average document length in the collection, k_1, k_3, and b are constants used for tuning. The k_1 parameter affects the term weight. If it is 0, then the term weight is reduced to its actual presence, meaning that the term weight is not affected by its frequency in the document, and if it is set to a larger value, the term weight increases as its frequency increases in the document. The tuning constant k_3 affects the number of term instances that contribute to the ranking. For example, if k_3 is set to 0, then only one instance of each query term contributes to the ranking. The constant b is used to control the document length normalisation. If it is set to 0, no normalisation will take place; if it is set to 1, then normalisation is in full effect. In TREC 6, the value of k_3 was 1.2, the value for k_3 was in the range from 0 to 1000 and the value of the b parameter was 0.75 [Walker et al., 1997]. These values have also been used in TREC 7 and TREC 8. Chowdhury et al. [2002] determined different values for the b parameter and showed significant improvements when setting this value to 0.25. He and Ounis [2005] proposed a method for tuning the term frequency normalisation parameter that is independent of any collection, and showed that their new tuning method achieves results that are at least as good as or significantly outperform the default settings of Okapi BM25 parameters. El-Khair [2003] conducted several unofficial runs to tune these parameters for the TREC 2001 Arabic collection, but his attempts did not improve the results over the initial parameters set for English. It is not clear what parameters he examined, nor what range of values were tested.

In our retrieval experiments in Chapter 4, we use the Okapi BM25 weighting model with default values determined for English (k_1= 1.2, k_3= 7, and b= 0.75). We determine new values for our Arabic text collections in Chapter 5.

Language Models

Liu and Croft [2005] define language modelling (LM) as "the task of estimating a probability distribution that captures statistical regularities of natural language use". Language modelling assumes that the query and the documents relevant to it are generated using the same language model. It has been used successfully in many applications, including speech recognition, machine translation, and spelling correction, and has been used in IR experiments by Ponte and Croft [1998]. In this retrieval model, documents are ranked by the likelihood that a document is ideal to generate the query. A statistical language model (SLM) computes the probability of all linguistic units (grams) in a language [Rosenfeld, 2000]. The aim of the SLM is to determine the likelihood that a gram would occur in the document, given the preceding gram in the document. Suppose that d represents a document that has n words w, then the probability of the document d is given by:

$$P(d) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_n|w_1w_2w_3\ldots w_{n-1}) = \prod_{i=1}^{n} P(w_i|w_1...w_{i-1}) \quad (2.7)$$

In the n-gram language model, probability is usually estimated using n-gram frequencies in a training data set. Some grams might not exist in the training data, and would cause a problem in estimating the probability of new unseen grams, since their probability would be zero, and the document probability is computed as a product of the n-gram probabilities (Equation 2.7). To address this problem, smoothing is used. This is usually done by increasing the lower probabilities and reducing the higher probabilities to make the overall probability equal to one [Liu and Croft, 2005].

Three different approaches are followed in LM. The first model assumes that the document model generates the query; the second model assigns probabilities to documents based on the likelihood that the query model generates the document; and in the third approach, a language model is developed for the query and compared with each document language model in the collection. Details of each approach are given by Liu and Croft [2005].

The Bayesian Inference Networks Probabilistic Model

This model is a directed acyclic graph built of nodes and edges. Nodes are either prepositional variables or constants, while edges are dependencies between nodes. A direct edge (arc) is drawn between two nodes if a proposition represented by one node implies another. The belief in a proposition between two nodes is represented by a value on the arc. The Bayesian model enables this value to be computed given the belief in its parent node. Given a set of

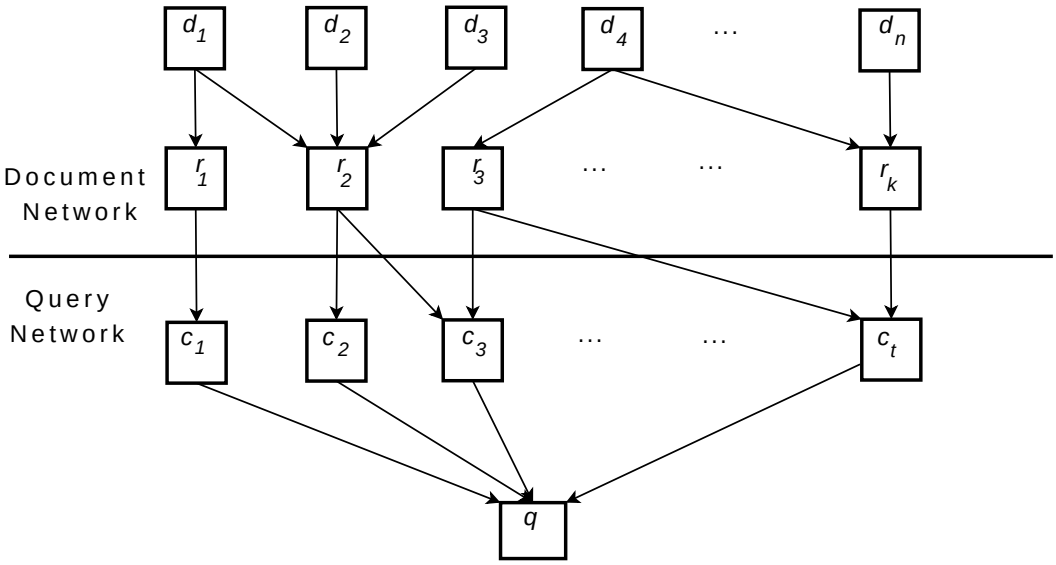

Figure 2.1: Document retrieval inference network model. The document network is composed of n documents with k content representations. The query model has one query with t concepts. Figure derived from [Callan et al., 1992].

values of prior probability assigned to the roots of this graph, the belief of other nodes in the graph can be computed [Turtle and Croft, 1990; 1991].

The Document Retrieval Inference Network: This is an instance of the Bayesian inference model that represents both the document collection and the query using two component networks. Figure 2.1 shows an example of this model with a document network that has two abstract levels, the document text level and the content representation level; and a query network with two abstract levels, the query level and the concept level [Callan et al., 1992].

In the document level, d_i nodes are roots with one or more content representation nodes r_k. Every document node is assigned a prior probability. This is usually $\frac{1}{n}$, where n is the number of documents in the collection. The dependency between the content representation nodes and document nodes is calculated using the conditional probability $P(r_k|d_i)$. The content representation nodes represent a proposition that a concept is seen. These nodes are connected with the query concept nodes in the query network.

The query network represents the user information needs. In our example, the query

node q represents a proposition that a user information need is met, and the concept node represents a proposition that a document contains the concept c. Nodes in the graph are either *true* or *false* except for the document and the query nodes, which are assigned the value *true*.

The INQUERY Retrieval System [Callan et al., 1992] constructs document networks using a straightforward mapping between documents and content representation nodes; this mapping is stored in an inverted file index to facilitate retrieval. Query networks are constructed by converting the natural language queries to structured queries. The system evaluates the root node of the query network and returns a list of documents and the value of the belief that they meet the query. INQUERY uses 9 operators to structure queries. For example, the #sum operator returns the average belief value for terms in its scope, while the #syn operator considers terms included in its scope as synonyms [Callan et al., 1992]. In Chapter 7, we use the INQUERY retrieval method to expand foreign words in queries using their variants.

We use the INQUERY retrieval model in testing query expansion in Chapter 7.

String and Phonetic Similarities

In the above section, we showed how to find similar documents to a user query. We now discuss similarity measures that can be used to compare strings.

One of the main issues in IR is to find proper nouns. Many writing conventions are used to write proper nouns, usually resulting in different spellings, but the same pronunciation. The problem becomes worse when names are transliterated from one language to another. For example, "ahmed", "ahmmed", and "ahmad" are three different versions for the Arabic name "أحمَد" /ʔħmad̪/. If one version is written in the query, search engines would fail to retrieve other versions without using some sort of weighting. In this section, we present techniques used to identify similar words based on their pronunciation and spelling.

Approaches to identify similar-sounding but differently-spelt words have been heavily investigated in English; among these are techniques that make use of string or phonetic similarity.

String similarity approaches include the Edit Distance [Hall and Dowling, 1980], used to measure the similarity of two strings by counting the minimal number of character insertions, deletions, or replacements needed to transform one string into another. To transpose a string s of length n into a string t of length m, $edit(n, m)$ computes the minimal steps required as

		a	h	m	m	e	d
	0 (0,0)	1 (1,0)	2 (2,0)	3 (3,0)	4 (4,0)	5 (5,0)	6 (6,0)
a	1 (0,1)	0 (1,1)	1 (2,1)	2 (3,1)	3 (4,1)	4 (5,1)	5 (6,1)
h	2 (0,2)	1 (1,2)	0 (2,2)	1 (3,2)	2 (4,2)	3 (5,2)	4 (6,2)
m	3 (0,3)	2 (1,3)	1 (2,3)	0 (3,3)	1 (4,3)	2 (5,3)	3 (6,3)
e	4 (0,4)	3 (1,4)	2 (2,4)	1 (3,4)	1 (4,4)	1 (5,4)	2 (6,4)
d	5 (0,5)	4 (1,5)	3 (2,5)	2 (3,5)	2 (4,5)	2 (5,5)	1 (6,5)

		k	h	a	l	e	d
	0 (0,0)	1 (1,0)	2 (2,0)	3 (3,0)	4 (4,0)	5 (5,0)	6 (6,0)
k	1 (0,1)	0 (1,1)	1 (2,1)	2 (3,1)	3 (4,1)	4 (5,1)	5 (6,1)
a	2 (0,2)	1 (1,2)	1 (2,2)	1 (3,2)	2 (4,2)	3 (5,2)	4 (6,2)
l	3 (0,3)	2 (1,3)	2 (2,3)	2 (3,3)	1 (4,3)	2 (5,3)	3 (6,3)
i	4 (0,4)	3 (1,4)	3 (2,4)	3 (3,4)	2 (4,4)	2 (5,4)	3 (6,4)
d	5 (0,5)	4 (1,5)	4 (2,5)	4 (3,5)	3 (4,5)	3 (5,5)	2 (6,5)

Figure 2.2: Calculating Edit Distance between the strings "ahmed" and "ahmmed" (left) and "kalid" and "khaled" (right). The final computed distances between the string pairs are the values in the bottom-right corner of the alignment matrix.

follows:

$$\begin{aligned} &edit(0,0) = 0 \\ &edit(i,0) = i \\ &edit(0,j) = j \\ &edit(i,j) = min[edit(i-1,j)+1, \\ &\qquad\qquad edit(i,j-1)+1, \\ &\qquad\qquad edit(i-1,j-1)+d(s_i,t_j)] \end{aligned} \tag{2.8}$$

where i indexes string s and ranges from 0 to n, and j indexes string t and ranges from 1 to m; and $d(s_i, t_j) = 0$ if $s_i = t_j$, and equals 1 otherwise. The algorithm starts by assigning the value 0 to the first position in the matrix (edit[0,0]), the i^{th} value to elements in the first row, and the j^{th} value to elements in the first column. Starting at position $edit[1,1]$ and ending at position $edit[m,n]$ the algorithm first computes the function $d(i,j)$ by comparing the i^{th} character in string s with the j^{th} character in string t. If they are equal, $d(s_i, t_j)$ equals 0, otherwise it is 1. The value of $edit[i,j]$ is computed by examining the elements to the top, left, and top-left according to Equation 2.8. For example, $d(s_1, t_1) = 0$ as s[1]="a", and t[1]="a". Accordingly $edit[1,1] = min(edit[0,1]+1, edit[1,0]+1, edit[0,0]+0) = 0$. The similarity (distance) between "ahmed" and "ahmmed" is 1, while it is 2 between the two words "khaled" and "kalid" (see Figure 2.2).

Another candidate approach that can be used to identify similar words is n-grams [Hall and Dowling, 1980]. This approach is language-independent; the strings are divided into grams (substrings) of length n, and the similarity of the strings is computed on the basis of the similarity of their n-grams.

Pfeifer et al. [1996] compute the similarity as the number of shared grams divided by the total number of distinct grams in the two strings,

$$gramCount = sim(s,t) = \frac{\mid G_s \cap G_t \mid}{\mid G_s \cup G_t \mid} \tag{2.9}$$

where G_s is the set of grams in string s, and G_t is the set of grams in string t. For example, with n=2, the similarity of "ahmed" and "ahmmed" using this measure is 0.8 because both strings contain the four 2-grams "ah", "hm", "me", and "ed", while there are five distinct 2-grams across the two strings.

Pirkola et al. [2002] tested the concept of skip grams (s-grams). These are formed by combining characters based on the number of skipped characters. For a word with n characters, the possible *Character Combination Index (CCI)* of skipped characters can be 0, 1, 2, ..., n-m where m is the gram size. For example, when using bigrams, the CCI=(0) for the word "grams" represents the set of bigrams with 0 skipped characters, which is {"gr","ra","am","ms"}. If CCI=(1), then the set of s-bigrams is {"ga","rm","as"}, and if CCI=(0,1), then set of s-bigrams is a combination of the previous two sets. Pirkola et al. [2002] used the same n-gram similarity measure used by Pfeifer et al. [1996] to compare words and their variants in English, Finish, German, and Swedish. They found that for short words, s-grams are more effective than conventional n-grams. In Chapter 7, we describe experiments using s-grams with CCI=(0,1), to match foreign word variants in Arabic.

Gram distance [Ukkonen, 1992] is another string similarity technique. When grams are not frequently repeated — which is the case in short strings such as names — the similarity is computed as [Zobel and Dart, 1996]:

$$gramDist(s,t) = \mid G_s \mid + \mid G_t \mid -2 \mid G_s \cap G_t \mid \tag{2.10}$$

According to this measure, the distance between "ahmed" and "ahmmed" is 1.

With the Dice measure [Dice, 1945], the similarity of strings s and t is computed as twice the number of common n-grams between s and t, divided by the total number of n-grams in the two strings:

$$Dice(s,t) = \frac{2\times \mid G_s \cap G_t \mid}{\mid G_s \mid + \mid G_t \mid} \tag{2.11}$$

Code	0	1	2	3	4	5	6	7	8	9
Soundex	a e i o u y h w	b f p v	c g j k q s x z	d t	l	m n	r			
Phonix	a e i o u y h w	b p	c g j k q	d t	l	m n	r	f v	s x z	
Editex	a e i o u y	b p	c k q	d t	l r	m n	g j	f p v	s x z	c s z

Table 2.10: Phonetic groups and their codes for English phonetic similarity algorithms.

The similarity between "ahmed" and "ahmmed" when using this measure is $\frac{8}{9} = 0.89$.

The longest common subsequence (LCS) algorithm measures the similarity between two strings based characters common to both strings [Wagner and Fischer, 1974; Stephen, 1992]. Similarity is normalised by dividing the length of the common subsequence by the length of the longer string [Melamed, 1995]. The similarity between "ahmed" and "ahmmed" is ($\frac{5}{6} = 0.833$).

Phonetic approaches to determine similarity between two words include the well-known Soundex algorithm developed by Odell and Russell, patented in 1918 and 1922 [Hall and Dowling, 1980]. This has predefined codes for the sounds in a language, with similar-sounding letters grouped under one code (see Table 2.10). During comparisons, all letters in a word bar the first are encoded, and the resulting representation is truncated to be at most four characters long. For example, "tareg", "tareq" and "tarek" are encoded to "T620". However, the algorithm has some flaws; some dissimilar-sounding strings, such as "catherine" and "cotroneo", are mapped to the same code, while some similar-sounding strings, such as "knight" and "night", are mapped to different codes [Zobel and Dart, 1996].

Enhancements to the Soundex algorithm have been made by manipulating strings before encoding, and by altering codes after encoding. Celko [2005] encoded strings using letters instead of numbers and used n-grams to substitute letters depending on their n-grams. For example, the letter "t" is replaced with "s" if it is found in the "nst" trigram. Letter substitution also depends on the position of the n-gram in the word. There are specific letter substitutions for prefixes, such as replacing the prefix "Mac" with "Mcc", and for suffixes such as replacing "nst" with "ns". The algorithm removes the letter "h" if it is preceded by "a" and delimits the new code using spaces. Holmes and McCabe [2002] used a similar n-grams substitution algorithm to replace letters in their n-grams. They used 25 rules to substitute the word n-grams. The new version of the word is then encoded using numbers as in the Russell Soundex, but different codes and groups are used. The algorithm is called Fuzzy Soundex. To address insertion and deletion errors that happen near the end

of the name, they used multiple phonetic codes generated by the Soundex algorithms, and to address the errors near the beginning of the name, they used the concept of *code shift* that removes the second letter of the five-bytes encoded strings. They also used the Dice measure to fuse results of different Soundex algorithms, and showed that integrating different algorithms increases recall to 96% with a precision of 70%.

A variant of Soundex is the Phonix algorithm [Gadd, 1990], which transforms letter groups to letters and then to codes; the actual groups are different from Soundex (see Table 2.10). Phonix applies a set of about 160 transformation rules to reduce strings to their canonical forms before encoding them. For example, the letters "cu" are replaced by "ku". Both Soundex and Phonix have been reported to have poorer precision in identifying variants of English names than both Edit Distance and n-grams [Zobel and Dart, 1995].

Editex, developed by Zobel and Dart [1996], enhances the Edit Distance technique by incorporating the letter-grouping strategy used by Soundex and Phonix. These groups are shown in Table 2.10. The algorithm has been shown to have better performance than Soundex and Phonix algorithms, as well as Edit Distance, on a collection of 30,000 distinct English names. The distance between two strings s and t is computed as:

$$\begin{aligned} &edit(0,0) = 0 \\ &edit(i,0) = edit(i-1,0) + d(s_i - 1, s_1) \\ &edit(0.j) = edit(0,j-1) + d(t_j - 1, t_j) \\ &edit(i.j) = min[edit(i-1,j) + d(s_i - 1, s_i), \\ &\qquad\qquad edit(i,j-1) + d(t_j - 1, t_j), \\ &\qquad\qquad edit(i-1,j-1) + r(s_i, t_j)] \end{aligned} \tag{2.12}$$

where i indexes string s and ranges from 0 to n, and j indexes string t and ranges from 1 to m; $r(s_i, t_j)$ is 0 if s_i=t_j, 1 if $group(s_i)$=$group(t_j)$, and 2 otherwise; and $d(s_i, t_j)$ is 1 if $s_i \neq t_j$ and s_i is "h" or "w", and $r(s_i, t_j)$ otherwise. Figure 2.3 shows how the distance between the string pairs "ahmed" and "ahmmed", and "kalid" and "khaled" is calculated using the Editex algorithm. The calculation is similar to Edit Distance. However, the algorithm uses another function that calculates the similarity between two characters based on their phonetic groups — $r(s_i, t_j)$. If a character at the i^{th} column from the s string and a character at j^{th} row from the t string are identical, then $r(s_i, t_j) = 0$, if they belong to the same phonetic group, then $r(s_i, t_j) = 1$, and it equals 2 otherwise. The algorithm considers the letters "h" and "w" as silent using the function $d(s_i, t_j)$ as shown in the recurrence relation in Equation 2.12.

		a	h	m	m	e	d
	0 (0,0)	2 (1,0)	4 (2,0)	5 (3,0)	5 (4,0)	7 (5,0)	9 (6,0)
a	2 (0,1)	0 (1,1)	2 (2,1)	3 (3,1)	3 (4,1)	5 (5,1)	7 (6,1)
h	4 (0,2)	2 (1,2)	0 (2,2)	1 (3,2)	1 (4,2)	3 (5,2)	5 (6,2)
m	5 (0,3)	3 (1,3)	1 (2,3)	0 (3,3)	0 (4,3)	2 (5,3)	4 (6,3)
e	7 (0,4)	5 (1,4)	3 (2,4)	2 (3,4)	2 (4,4)	0 (5,4)	2 (6,4)
d	9 (0,5)	7 (1,5)	5 (2,5)	4 (3,5)	4 (4,5)	2 (5,5)	**0** (6,5)

		k	h	a	l	e	d
	0 (0,0)	2 (1,0)	4 (2,0)	5 (3,0)	7 (4,0)	9 (5,0)	11 (6,0)
k	2 (0,1)	0 (1,1)	2 (2,1)	3 (3,1)	5 (4,1)	7 (5,1)	9 (6,1)
a	4 (0,2)	2 (1,2)	2 (2,2)	2 (3,2)	4 (4,2)	6 (5,2)	8 (6,2)
l	6 (0,3)	4 (1,3)	4 (2,3)	4 (3,3)	2 (4,3)	4 (5,3)	6 (6,3)
e	8 (0,4)	6 (1,4)	6 (2,4)	6 (3,4)	4 (4,4)	2 (5,4)	4 (6,4)
d	10 (0,5)	8 (1,5)	8 (2,5)	8 (3,5)	6 (4,5)	4 (5,5)	2 (6,5)

Figure 2.3: Calculating Editex distance between the strings "ahmed" and "ahmmed" (left) and "kalid" and "khaled" (right). The final computed distances between the string pairs are the values in the bottom-right corner of the alignment matrix.

In Chapter 7, we use the Edit Distance, Gram Count, Gram Distance, Dice, LCS and s-gram algorithms; and modify the Soundex, and Editex algorithms to accommodate matching transliterated foreign words in Arabic text.

2.2.4 Relevance Feedback

Relevance feedback, first described by Rocchio [1971], is a well-known technique to improve retrieval effectiveness in monolingual information retrieval [Salton and Buckley, 1990]. The idea behind relevance feedback is to expand user query with terms from relevant documents returned by running the initial query. In the first round, the user specifies which returned documents are relevant. Terms in those documents are then used by the retrieval system to expand the original query. This process can be repeated more than once until the user feels satisfied with the returned results. While Buckley et al. [1994] show that such an approach leads to a 19% to 38% increase in effectiveness depending on the number of relevant documents used, users are generally reluctant to provide feedback on returned documents [Dennis et al., 1998].

Another approach where queries are expanded automatically without the need for user intervention is called *pseudo relevance feedback* (also called automatic, blind or ad-hod relevance feedback). In this approach, queries are expanded using terms from the top-ranked retrieved documents, which the retrieval system usually assumes to be relevant.

Aljlayl [2002] used relevance feedback to test the effectiveness of light stemming for retrieving Arabic documents. The relevance feedback resulted in a 16% increase in the light stemming effectiveness, and a 71% increase over the baseline (no stemming). Using the TREC 2001 dataset, he determined that using the top 10 terms from the top 15 retrieved documents gives the best result. He suggested that the number of terms that give effective performance ranges from 10 to 20. Darwish et al. [2005] also used pseudo relevance feedback combined with a light stemmer, a morphological analyser, and context-based morphological analysers and showed that this resulted in a 6% increase in mean average precision. We use pseudo relevance feedback in Chapters 4 and 5 to evaluate its effects on light stemming when using morphological rules.

2.2.5 Cross-Lingual Information Retrieval

The growth in internet users worldwide has been accompanied by an increasing proportion of content in languages other than English. For example, according the Internet world statistics,[4] the number of Internet users in regions such as the Middle East, Africa, Asia, and Latin America has grown significantly more than the worldwide average. The need to search general Internet content, not only the portion in one's native language, led to the introduction of Cross-Lingual Information retrieval (CLIR) research. CLIR aims to bridge the gap between users and content by allowing queries in one language to be used to retrieve content in another. CLIR was first defined under Multilingual Information Retrieval (MLIR) by Hull and Grefenstette [1996]. In the same year, TREC initiated a CLIR track for English and other languages such as German, French, Spanish, and Dutch [Voorhees and Harman, 1997]. One of the newer fora is the Japanese National Institute of Informatics (NII) workshop on Japanese CLIR, which provide the NII Test Collection for IR Systems (NTCIR);[5] this collection includes data for the Chinese and Korean languages. In the year 2000, the Cross-Language Evaluation Forum (CLEF)[6] has also started a CLIR track on European languages, and later included other languages such as Amharic, Hindi, Indonesian and Arabic.

Measuring the performance of IR systems in CLIR tasks is similar to the normal IR retrieval tasks. Results are expected to be lower than typical for monolingual retrieval. We discuss evaluating IR in Section 2.3.

To search documents that are not in the same language as the query, we translate either

[4] http://www.internetworldstats.com
[5] http://research.nii.ac.jp/ntcir/index-en.html
[6] http://www.clef-campaign.org

the query or the entire collection. Translating documents is costly in terms of time and space, but the quality of translation is far better than when translating queries due to the greater amount of context available [Hull and Grefenstette, 1996]. Nevertheless, it is more tractable to translate the queries, and so it is the norm in CLIR.

With static collections, it is conceivable that documents be translated manually, however tedious that may be. However, in large, fast-growing, and dynamic collections such as the Web, such manual translation is infeasible, and we must rely on automated translation. Machine translation (MT) is the simplest form of automatic translation. A system accepts words in one language and produces a translation in the target language. Language-dependent rules are applied to produce syntactic sentences. OOV words such as proper nouns are usually transliterated using phonetic matching across the languages. Other automatic machine translation approaches use parallel corpora and statistical methods. Statistical Machine Translation (SMT) is a rapidly growing area of research that has resulted in systems that outperform commercial systems for some languages pairs such Arabic-English and Chinese-English [Koehn and Monz, 2006].

There are several automatic machine translation engines available on the Web. Some of these, particularly those capable of translating English to Arabic, are *AlMisbar*,[7] *Google Translate*,[8] and *Systran*.[9] In Chapter 4, we describe experiments that use these tools to translate queries.

As our focus in this thesis is neither CLIR nor MT, we do not explore this topic in further depth.

2.2.6 An Application Example: Video Retrieval

Finding videos has become one of the most popular search activities on the Web. In 2006 for example, the BBC reported that the word "video" was the seventh most-common search term entered into the Google search engine [BBC News, 2006]. For this reason, video retrieval has become a concern for commercial video companies and search engines.

TREC started a track on video retrieval in 2001. The focus of the track was to promote research in automatic segmentation, indexing, and content-based retrieval of digital video [Voorhees, 2001]. In 2003, the track became an independent evaluation under the name TRECVID. The main tasks initiated in TRECVID include shot-boundary detection,

[7] http://www.almisbar.com
[8] http://translate.google.com
[9] http://www.systransoft.com

that has been discontinued in 2008; and video segment retrieval. While the former task requires analysis of the visual content of the video, the latter can be approached using text generated by an Automatic Speech Recognition (ASR) system; these transcripts are aligned with the corresponding shots in the video stream, perhaps including one or two shots on either side to allow for gaps in speech and speed variations [Volkmer and Tahaghoghi, 2005]. Systems return a list of shots relevant to a particular information need.

Video retrieval performance is evaluated using normal IR techniques, such as the precision and recall techniques we describe in Section 2.3.

The TRECVID 2005 data set contains recorded television broadcast news in three languages — Arabic, Chinese, and English — with the associated ASR transcripts available [Over et al., 2006]. Of the 169 hours of footage, 43 hours are in Arabic, 52 hours are in Chinese, and 74 hours are in US English. Arabic and Chinese ASR collections are automatically translated to English to allow searching the whole collection in English. The collection has 24 English-language queries to be used to find specific video footage in the entire collection. The queries all begin with the phrase, "find shots of", and aim to find scenes containing a specific person, place or object, or a general view, building, or action.

In Chapter 4, we use the TRECVID 2005 collection to check the effectiveness of techniques used in normal AIR systems on ASR text.

2.2.7 Summary

In this section, we have described how information retrieval systems parse and index documents, and how they retrieve relevant documents in response to a query. During parsing many techniques are employed in order to extract the proper tokens from text. Words are normalised and highly frequent words are removed. Extracted terms are then indexed in a way that reflects their position and frequency in the text collection. To search the text collection, queries are parsed to extract terms that are then compared to information in the index about every document in the collection. We have explained several models proposed for this comparison, each with a different way of computing the similarity of a document to a query, and therefore to the user's information need. We have also introduced cross-lingual information retrieval (CLIR), and noted one application of Arabic CLIR explored by the research community. We follow with a discussion of techniques to evaluate the effectiveness of competing information retrieval approaches.

<DOC>
<DOCNO>19940520_AFP_ARB.0013</DOCNO>
<HEADER>
ارا٠٠٢٧ ٩٧٠٠ قبرص /افبتظذ ٤٤ كرة قدم
</HEADER>
<BODY>
<HEADLINE>
بطولة تونس: تعادل حمام الانف وشبيبة القيروان صفرصفر
</HEADLINE>
<TEXT>
تونس ٥-٠٢ تعادل حمام الانف وشبيبة القيروان صفرصفر مساء امس الخميس فى مباراة متاخرة ضمن بطولة تونس في كرة القدم.
ويحتل الفريقان المركز التاسع ولكل منهما ٩١ نقطة من ٣٢ مباراة. وكان الترجي التونسي ٧٣ نقطة ضمن احراز اللّقب قبل يومين
لانه يتقدم على منافسه المباشر النادي الصفاقسي بفارق سبع نقاط قبل ان تختتم المسابقة بثلاث مراحل.
</TEXT>
<FOOTER>
ـــــــــــــــــــــــــــــــ عل/سعد/م فسب افب
</FOOTER>
</BODY>
<TRAILER>
٢٠٩٠٠٢ ٤٩ جمت ماي
</TRAILER>
</DOC>

Figure 2.4: A sample document from the TREC 2001 collection [Gey and Oard, 2001].

2.3 Evaluation of IR Systems

The performance of an IR system is usually measured by its ability to find documents relevant to a query posed by a user (effectiveness), and how fast it is in doing so (efficiency). In this section, we describe how IR systems are measured using test collections and measures.

While some researchers have evaluated the effectiveness of IR systems by measuring user satisfaction [Spink, 2002; Al-Maskari et al., 2007], it is more common to examine how well a system performs on queries with known relevant answers (the ground truth). A set of queries with known relevant documents in a collection are run against the same collection using different systems. Results of each system are compared with the manually judged results and retrieval effectiveness is determined using measures such as those we describe below.

2.3.1 Test Collections and Evaluation Forums

To evaluate an IR system, we require a testbed with three main components: a data collection, comprising the text, image, or other documents to be searched; a set of queries that prescribe information needs that must be met; and a set of relevance judgments that lists the set of documents relevant to each query.

Some of the more widely used test collections used in IR research have been developed as part of the NIST Text Retrieval Conference (TREC) series.[10] Since 1992, the TREC series has explored different aspects of IR in various tracks, and has provided appropriate test collections and recommended evaluation methods [Voorhees, 2001]. Long-running tracks include the ad hoc search track, where the performance of a system is tested using a static set of documents and new search topics; the question-answering track, where systems must find answers to set questions [Voorhees, 2003]; and the cross-lingual track, where systems are provided queries in one language, and must return relevant documents in another language [Voorhees, 2001]. A detailed overview of the TREC tracks appears elsewhere [Voorhees, 2001].

As noted in Section 2.2.5, CLEF also explores collections and metrics for monolingual and cross-lingual information retrieval, though it focuses primarily on European languages; and NTCIR explores similar collections for Asian languages.

Building Test Collections

As mentioned earlier, a test collection has three main parts: a set of documents, a set of queries, and relevance judgements.

To evaluate Arabic text retrieval approaches, we collect text documents from sources such as web pages or newswire dispatches. Each document is associated with a unique identifier, and may be marked up using HTML or SGML tags. Figure 2.4 shows an Arabic document from the TREC 2001 collection; here, the `DOC` tags indicate the limits of the document, while the `DOCNO` tags enclose the unique document identifier. Many document collections have been used by TREC. These include newswire document collections such as Agence France Press (AFP) Arabic Newswire [Gey and Oard, 2001], and documents crawled from the Web such as WT10g collections used in the web track in TREC 9 [Voorhees and Harman, 2000].

Queries — also called "topics" in TREC — have special SGML markup tags. The left side of Figure 2.5 shows a sample query from the TREC 2001 Arabic collection. As with

[10] http://trec.nist.gov

<top>
<num> Number: 8
<title> المسرح في مصر
<desc> Description:
ماذا قالت الصحف عن المسرح في مصر
<narr> Narrative:
يتعلق بالموضوع كل المقالات التي تحدث عن الساحة
المسرحية في مصر في الماضي و في الحاضر
</top>

QID		DocID	Rel
1	0	19940515_AFP_ARB.0095	0
1	0	19940519_AFP_ARB.0085	0
1	0	19940526_AFP_ARB.0068	1
...			
2	0	19950518_AFP_ARB.0134	1
2	0	19950521_AFP_ARB.0048	0
...			
3	0	19950115_AFP_ARB.0215	1
...			

Figure 2.5: A sample topic taken from the TREC 2001 collection (left), and a sample relevance judgements (right), "QID" stands for query number, "DocID" stands for document identifier, and "Rel" stands for relevance. Note that tags are not necessarily paired.

documents, queries have unique identifiers. This is indicated using the tag <num>. Three tags are used in TREC queries to indicate the user information need. The <title> text represents a short query that might be typed in by a user. The <desc> text clarifies the information need; for example, a query title may be "cats", which is rather broad, but the description "where is the musical Cats playing?" clarifies the specific information need. The <narr> text gives a longer explanation than the <desc> field. The aim of including both the description and the narrative is to test the effectiveness of longer queries and to clarify the user information need. They also serve as constant guidelines for assessors who judge the relevance of documents to a particular topic. The third part of a test collection is *relevance judgements*. In fact, this is what turns the set of documents and the set of queries into a test collection [Voorhees, 2001]. For every query, each document in the collection is either marked as relevant or not relevant. The right side of Figure 2.5 shows a sample of relevance judgements. The first column is the topic ID, the second is an unused field usually set to 0, the third is the document ID, and the last is the relevance column; 0 indicates that a document is not relevant to a topic, while 1 indicates relevance. Drawing up relevance judgments for a collection requires human input, and is both tedious and costly.

With small collections, it is possible to form thorough judgments for all (query, document) pairs [Zobel, 1998]; however, this is infeasible with the much larger collections used in IR research today. For example, a collection with 800,000 documents requires over 6,500 hours to completely judge all documents for one query, assuming that 30 seconds are required to judge a single document [Voorhees, 2001]. To minimise the effort needed to judge large collections, some techniques have been developed, the best-known of which is *pooling* [Jones

and van Rijsbergen, 1975], which operates on the premise that almost all relevant documents will be ranked highly by one or another of multiple IR systems, and that we can approximate complete relevance judgments by simply pooling the N top-ranked results from each system and assessing these alone. In TREC, the first 100 documents of each run are added to the pool [Voorhees, 2001]. Zobel [1998] reports that pooling captures some 50-70% of all relevant documents, and that it is a reliable technique; Sanderson and Zobel [2005] add that shallow pools lead to more reliable judgments.

An alternative technique used to create the ground truth with minimal human effort is *Interactive Searching and Judging (ISJ)* [Cormack et al., 1998]. The aim of this technique is to produce relevance judgements with minimal human effort. In this techniques, assessors aim to find as many relevant documents as possible for a query, and can reformulate queries as required until they conclude that further relevant documents are unlikely to be returned by the system. Cormack et al. [1998] showed that this method produces similar results to pooling for the TREC 6 collection.

Using 121 search results submitted to the first NTCIR Workshop, Kuriyama et al. [2002] showed that pooling with top 100 documents (P100) captures 89.2% of relevant documents for topics with 50 or fewer relevant documents. In an attempt to capture documents relevant to topics with more than 100 relevant documents, they showed that ISJ is more effective than P100 and automatic runs. Sanderson and Joho [2004] have considered TREC 5, 6, 7, and 8 manual runs as ISJ runs and compared their performance with TREC relevance judgements. They concluded that the method is "broadly applicable regardless of retrieval system used or people employed to conduct the searching process", and that this method can be used to form a test collection quickly and with limited resources.

In Chapter 5 we use this method for building a new test collection for Arabic.

2.3.2 Arabic TREC 2001 and 2002 testbed

A collection specifically designed to evaluate Arabic text information retrieval systems was created as part of TREC 2001. The collection has 383,872 Arabic documents, mainly newswire dispatches published by Agence France Press (AFP) between 1994 and 2000. Standard TREC queries and ground truth have been generated for this collection: 25 queries were defined as part of TREC 2001, and 50 additional queries were developed for TREC 2002. Both sets of queries have corresponding relevance judgements created using the pooling technique. In TREC 2001 the pool was formed using the top 70 ranked documents of 30 runs

submitted by ten research teams: 15 cross-lingual runs with English queries, 1 cross-lingual run with French queries, and 14 monolingual runs with Arabic queries. Duplicate documents were removed from the pool, and documents were ordered in their canonical order to allow fair judgement by the user who originally wrote each topic. The average number of relevant documents per topic is 165. There has been some criticism of bias in these judgments. Gey and Oard [2001] point out that the topics have unusually long titles; that for 7 topics out of the 25, most relevant documents — more than half — were retrieved by only one participating system; and that for another 6 topics, 40 to 50% of the relevant documents were retrieved in the top 70 by only one system. They conclude that while this collection can be used for tuning, it is less useful for comparative studies.

TREC 2002 avoided the first problem by ensuring that no one group contributed more than 6% of the relevant judgments. Based on the results obtained by participants in TREC 2002, Oard and Gey [2002] suggest that the TREC 2002 topic are suitable for post-hoc use by automatic systems that did not contribute to the pool; they also recommend that the TREC 2002 topics be kept distinct from the TREC 2001 ones.

2.3.3 Measuring Effectiveness

To evaluate the effectiveness of an IR system, we assess how well it ranks documents relevant to a set of queries above documents that are not relevant. In this section we review the main measures used to evaluate IR systems.

Figure 2.6(a) shows the first fifteen documents returned by an IR system for the query Q8. The corresponding relevance judgments for Q8 are shown in Figure 2.6 (b). We continue with an explanation of the most common measures used for IR retrieval performance: recall, and precision [Witten et al., 1999].

Recall

Recall measures the ability of a system in retrieving all documents relevant to a query [van Rijsbergen, 1975]:

$$Recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents\ in\ the\ collection} \tag{2.13}$$

In Figure 2.6 (b), eleven documents have been judged a priori to be relevant to this query, but the system "S" has retrieved only six of these in its first fifteen results. The resulting

Rank	DocID	Score	
1	ARB20	92.8605	0
2	ARB15	92.0397	1
3	ARB28	82.9158	0
4	ARB01	77.7094	0
5	ARB04	77.0358	0
6	ARB17	75.1238	0
7	ARB23	73.6085	1
8	ARB29	72.9937	0
9	ARB27	72.8858	0
10	ARB03	70.6121	1
11	ARB22	68.0985	1
12	ARB16	67.6973	1
13	ARB11	67.0887	1
14	ARB10	64.0923	0
15	ARB18	63.8482	0

(a) Results for Query "Q8"

QID	DocID	Rel	QID	DocID	Rel
Q8	ARB01	0	Q8	ARB16	1
Q8	ARB02	1	Q8	ARB17	0
Q8	ARB03	1	Q8	ARB18	0
Q8	ARB04	0	Q8	ARB19	0
Q8	ARB05	0	Q8	ARB20	0
Q8	ARB06	0	Q8	ARB21	0
Q8	ARB07	1	Q8	ARB22	1
Q8	ARB08	0	Q8	ARB23	1
Q8	ARB09	0	Q8	ARB24	0
Q8	ARB10	0	Q8	ARB25	1
Q8	ARB11	1	Q8	ARB26	0
Q8	ARB12	1	Q8	ARB27	0
Q8	ARB13	1	Q8	ARB28	0
Q8	ARB14	0	Q8	ARB29	0
Q8	ARB15	1	Q8	ARB30	0

(b) Relevance for Query "Q8"

Figure 2.6: Retrieved document ranked by their relevance to query "Q8". "0" indicates that a document is not relevant, and "1" indicates relevance. The ranking is taken from a real IR experiment, but relevance is hypothetical.

recall at fifteen documents returned is $\frac{6}{11} = 0.545$. Similarly, the recall at ten documents returned is $\frac{3}{11} = 0.273$. Overall recall is typically measured at 1000 documents returned.

Precision

Precision is the fraction of the retrieved documents that are relevant to the query [van Rijsbergen, 1975]:

$$Precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ retrieved\ documents} \tag{2.14}$$

Back to our example, the precision of the system "S" at the cutoff value 15 is $\frac{6}{15} = 0.4$, and at cutoff 10 it is $\frac{3}{10} = 0.333$. Precision of IR systems is typically reported for cutoff values 5, 10, 20, or 100. Precision at ten results returned is very important as 85% of users examine only one page of results (typically the top ten retrieved documents) [Henzinger, 2000]. This

indicates the importance of precision at cutoff value 10, represented as "P@10", that we report throughout the thesis.

A variant of this measure is R-Precision, which is precision at rank R, where R is the number of relevant documents in the collection. In our example, we have 11 relevant documents, thus R-Precision is $\frac{4}{11} = 0.364$. The problem with R-Precision is that its typical value does not indicate the actual value of recall, as since some of the relevant documents may exist after the R^{th} rank.

Average precision (AP) is used to compute the average precision over all ranks in the answer set. Precision is calculated after every relevant document is found. Based on our example, relevant documents are found in ranks 2, 7, 10, 11, 12, and 13, therefore the precision values at these points are, $\frac{1}{2}$, $\frac{2}{7}$, $\frac{3}{10}$, $\frac{4}{11}$, $\frac{5}{12}$, $\frac{6}{13}$ respectively. The average precision is calculated by dividing the sum of the precisions at the different points by the number of relevant documents as follows:

$$AP = \frac{0.5 + 0.286 + 0.333 + 0.364 + 0.417 + 0.462}{11} = 0.215$$

This measure is more useful with ranked results than the previous measures. For example, in our running example, the P@10 would remain unchanged at $\frac{3}{10}$ whether the three relevant documents are the top three or bottom three. However, AP would drop from 0.273 to 0.101. Mean Average Precision (MAP) is the average AP score over set of queries. We use MAP to evaluate all our retrieval experiments.

Another measure, used to evaluate the precision when the first relevant document is retrieved, is *mean reciprocal rank (MRR)*. In our example, the reciprocal rank is $\frac{1}{2}$ as the first relevant document is found in rank 2. The reciprocal rank is calculated for all queries, then they are averaged to obtain the mean. This measure is also sensitive to ranking and used mainly to evaluate systems that are required to retrieve one answer to a particular query, such as question-answering tasks [Corrada-Emmanuel and Croft, 2004].

Probability of Relevance

Results of IR systems are usually ordered by a similarity value called *Retrieval Status Value* (RSV), shown in the second column of Figure 2.6 (a). This is usually calculated by ranking algorithms such as the cosine or probabilistic models discussed in the previous section. If the ranking algorithm is perfect, it produces a *linear ordering* — each document has a unique RSV. However, in the frequent case that two documents are assigned equal RSVs, they are arbitrarily placed one after another in a *weak ordering* [Raghavan et al., 1989]. Precision and

Doc ID	Score	Rank	
ARB20	92		0
ARB15	92	R_1	1
ARB28	92		0
ARB01	77		0
ARB04	77	R_2	0
ARB17	75	R_3	0
ARB23	72		1
ARB29	72	R_4	0
ARB27	72		0
ARB03	70	R_5	1
ARB08	68	R_6	0
ARB16	67		1
ARB02	67	R_7	0
ARB10	67		0
ARB18	63	R_8	0

Table 2.11: An example of weak ordering, where some documents have identical similarity scores. Normal precision and recall measures are not reliable with this ordering as it is possible for a relevant document to be retrieved in another position in the same rank.

recall are not reliable measures for weak ordering, due to the many possible permutations of documents that have equal RSVs. Raghavan et al. [1989] propose that the precision instead be represented by the probability that a retrieved document (ret) is relevant (rel):

$$P(rel|ret) = \frac{P(rel \cap ret)}{P(ret)} \tag{2.15}$$

The probability that a document is retrieved in a rank with n documents is calculated as:

$$P(ret) = \sum_{i=0}^{n} P(rel|arrangement_i)P(arrangement_i) \tag{2.16}$$

Let r be the number of retrieved documents across all ranks, and let nr be the number of non-relevant documents retrieved in an $arrangement_v$ in order to get t relevant documents. Thus, the probability of retrieving documents in that particular arrangement is given by:

$$P(ret) = P(rel|arrangement_i) = \frac{nr + t}{r} \tag{2.17}$$

The precision of retrieving one relevant document from the whole list in Table 2.11 is $\frac{1}{15}$, and since the probability of retrieving one document from the first rank in all arrangements is $\frac{1}{3}$, therefore,

$$P(ret|arrangement_0)P(arrangement_0) = \frac{0+1}{15} \cdot \frac{1}{3} = \frac{1}{45},$$

$$P(ret|arrangement_1)P(arrangement_1) = \frac{1+1}{15} \cdot \frac{1}{3} = \frac{2}{45},$$

and

$$P(ret|arrangement_2)P(arrangement_2) = \frac{2+1}{15} \cdot \frac{1}{3} = \frac{3}{45}.$$

Based on the above calculation, then

$$\begin{aligned} P(ret) &= \frac{1}{45} + \frac{2}{45} + \frac{3}{45} \\ &= \frac{6}{45} \end{aligned}$$

The final precision is then calculated by substituting these values in Equation 2.15

$$P(rel|ret) = \frac{\frac{1}{15}}{\frac{6}{45}} = 0.5$$

This measure is called the *probability of relevance (PRR).*

Assume that we want to retrieve NR relevant documents. We start at the first rank and go down until we find the last relevant document NR^{th} document at rank k. To calculate the PRR at a particular NR^{th} relevant document (recall), Raghavan et al. [1989] derived the following equation:

$$PRR = \frac{NR}{NR + j + (i.s)/(r+1)} \tag{2.18}$$

where NR is the number of relevant documents required, j is the number of non-relevant documents found in ranks before k, s is the number of remaining relevant documents still to be retrieved in rank k, i is the number of non-relevant documents in rank k, and r is the number of relevant documents in rank k.

To smooth results and average multiple queries, interpolation is used. Different queries have different numbers of relevant documents. This results in different recall points for different queries. For example, in Table 2.11, normal recall points are 1/11, 2/11,... 11/11. However, these points might not be the same for a query which has 20 relevant documents (1/20, 2/20, ... 20/20). To solve this problem, PRR is calculated at fixed recall points for all queries, and then interpolated to fixed recall points. Raghavan et al. [1989] have

also proposed two other measures: *Expected Precision (EP)* and *PRECALL*. Each produces different values than the PRR. Raghavan et al. [1989] showed that results produced by PRR and EP are more consistent than PRECALL.

Another approach to evaluate weak ordering is suggested by Zobel and Dart [1996], in which they shuffle weak ranks to generate random permutations and then calculate the average precision over ten permutations. Holmes and McCabe [2002] re-rank weak-ordered ranks using the Dice co-efficient to produce a linear ranking and then calculate precision and recall values.

We use PRR to evaluate algorithms that return weak ordering results in Chapter 7.

Combining Precision and Recall

In cases where one system achieves better recall than another but has lower precision, or vice versa, a harmonic measure that combines these two measures into one single value might provide a better evaluation. The *F-measure* is one of these measures which combines precision and recall [Jardine and van Rijsbergen, 1971]. A balanced version is called the F_1-measure (also known as the harmonic F-measure), and is computed as:

$$F_1(recall, precision) = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.19}$$

We use this measure in Chapter 6 to compare the effectiveness of identifying foreign words in Arabic text.

2.3.4 Measuring Efficiency

The efficiency of IR systems is usually measured in terms of processing time and memory requirements. Stemmers conflate terms, and so reduce index size; the degree of reduction is dependent on how aggressive the stemmer is. We report index size and processing time in Chapters 4, 5, and 7 when investigating different stemming and similarity matching techniques.

2.3.5 How Effective are New Algorithms?

Zobel [1998] stated that in many cases a system that shows an improvement over another system is not necessarily better, and recommended that the Wilcoxon signed-rank test [Wilcoxon, 1945] is a reliable indicator of significance for information retrieval. However, Smucker et al. [2007] report that the Wilcoxon signed-rank test and the sign test incor-

rectly predict significance, and that IR researchers should avoid using these tests; they also conclude that the t-test [Hull, 1993] can be used to evaluate the significance of differences in means. Accordingly, we use the t-test to evaluate significance in our experiments. Therefore, all p-values reported in this thesis are calculated using the t-test. We indicate significance in our results using the sign "$\uparrow$" if an improvement above the baseline is at the confidence level of 95% ($p < 0.05$), "$\Uparrow$" if it is at the 99% confidence level ($p < 0.01$), and "$\downarrow$" if it is significantly worse than the baseline at the 95% confidence level ($p < 0.05$).

2.3.6 Tools used in IR Evaluation

To facilitate research on IR, many tools have been developed to conduct IR experiments and evaluate systems. In this section, we briefly describe tools the we use throughout the thesis to evaluate IR experiments.

The Lemur Toolkit is an open-source toolkit designed to facilitate research in language modeling and information retrieval.[11] It supports indexing and searching several types of text collections including text documents written in Arabic CP1256 encoding. Latter releases include a search engine called "Indri" that is capable of indexing UTF-8 text documents. We use this toolkit to run all our retrieval experiments.

The toolkit indexes text collections and facilitates searching them using a list of topics. Using several retrieval models such as the vector space model and the BM25 Okapi model, the toolkit compares topics with documents and retrieves a list of ranked documents for every topic.

To evaluate precision and recall for every topic, another tools are used. We use the NIST `trec_eval` application to evaluate the returned lists against the relevance judgements for the text collection[2]. The application accepts both the relevance judgement file — called qrels — and the Lemur result files and outputs the precision and recall measures.

To calculate the PRR measure, we used a perl script developed by Norbert Gövert[3]. This script has been used by participants in the "INitiative for the Evaluation of XML Retrieval (INEX)" in 2004 to evaluate their systems.

All statistical significance tests are evaluated using the R statistical package.[4] The package is an open-source that has the capability for statistical computing and graphics. It is

[11] http://www.lemurproject.org
[2] http://trec.nist.gov/trec_eval/
[3] http://search.cpan.org/~goevert/RePrec-0.032/lib/RePrec/PRR.pm
[4] http://www.r-project.org/

developed at Bell Laboratories by John Chambers and colleagues. The package runs on different platforms including windows and unix.

2.3.7 Summary

In the preceding section, we have presented common approaches to evaluating IR systems, including creating static document collections and developing ground truth using human judgements. We have described the pooling and ISJ methods for reducing the judgment load, and noted that while the latter has not been as widely used in IR experiments, it is reported to lead to judgments as reliable as those obtained through pooling. We also described metrics for evaluating the effectiveness and efficiency of an IR system, and discussed how the PRR measure can be used for weakly-ordered results where the traditional measures of precision and recall may produce unreliable results.

2.4 Chapter Summary

In this chapter, we have presented background information about the Arabic language, its morphology and grammar. We have also described techniques used in the IR community to improve and test retrieval effectiveness, and presented a review of major contributions to Arabic Information Retrieval (AIR) research.

In Section 2.1 we have described the Arabic language, orthography, grammar, and morphology. Arabic uses a different style of writing than English and other Latin languages. The language has 28 letters and eight short vowels indicated using diacritics. It is written from right to left, and letters are usually attached to each other to form words. Letters can have up to four different shapes according to their position in the word. Arabic words are either nouns, verbs, or particles. Arabic words are also coined based on the concept of dual and femininity — concepts that are not found in English. Arabic affixes are categorised as common, noun, or verbal affixes. We have presented rules that govern how affixes may be attached to words. We have also described how foreign words may take on different variants when transliterated into Arabic script.

In Section 2.2, we have described information retrieval systems, and how competing IR systems can be evaluated. We have explained how terms are parsed, normalised, stopped, stemmed, tokenised, and indexed. We have described the theory of IR in general and how IR systems are evaluated. Parsing has been explained and an example has been given to illustrate term extraction, normalisation, stopping, stemming, and tokenisation.

We have also explained the way IR systems search indexing terms, and how these systems determine similarity between the query and the documents. Phonetic and string similarity techniques used to measure similarity between strings are also presented.

In Section 2.3, we have described how IR systems are evaluated, and how the ground truth for test collections can be derived using pooling or ISJ. We have explained the different measures used by the IR community to evaluate both retrieval effectiveness and efficiency. We defined precision as the proportion of relevant documents among the documents retrieved, and recall as the proportion of relevant documents in the collection that are retrieved by the system. If IR systems retrieve documents in the same rank, a weak ordering occurs. In such a case, precision and recall give unreliable results and another measure called PRR can be used. We have also presented the efficiency measures used in IR, and statistical tests used to determine the significance of improvements in IR systems.

We continue with a review of prior work in the field of Arabic text information retrieval.

Chapter 3

Arabic Information Retrieval

In this chapter we review Arabic Information Retrieval (AIR) systems, techniques used to find name variants, and possible approaches that can be used to distinguish foreign words from native words.

3.1 Arabic Information Retrieval Systems

We describe AIR systems under three broad categories: morphological analysers, light stemmers, and statistical approaches. Morphological analysers attempt to identify the affixes, stem, and root of a given word, and are primarily used for natural language processing (NLP) tasks such as part-of-speech tagging. In contrast, light stemmers focus on removing affixes to improve retrieval effectiveness, and do not attempt to identify grammatically correct stems. Finally, statistical approaches extract n-grams for indexing and retrieval, and operate independently of any language-specific rules.

3.1.1 Morphological Analysers

Early researchers were influenced by the traditional way of indexing Arabic text using root words, and developed systems based on morphological analysis and root extraction. Most of these systems have not been tested using standard IR evaluation techniques [Larkey et al., 2007].

El-Sadany and Hashish [1989] developed a morphological analyser that deals with vowelised, semi-vowelised, and fully vowelised text. The system accepts a word and returns its different morphological characteristics, such as the vowelised version, the root, and the pattern. The system has also the capability to accept a sentence typed by a user and to provide

vowelised versions of the words in that sentence; it also allows the user to clarify ambiguity in the sentences. No evaluation has been provided for this system.

Al-Fedaghi and Al-Anzi [1989] developed a system to find the triliteral root of Arabic words. The system has two lists: a list of Arabic patterns and a list of valid triliteral Arabic roots. The pattern list contains not only the basic Arabic patterns, but also patterns with valid affixes attached to them. Rather than remove affixes, the system compares input words with patterns of the same length, and returns the corresponding root if it exists in the valid root-word list. The authors report that their algorithm successfully extracts roots for up to 80% of the words in a small text collection; however, no accuracy figures are reported [Khoja and Garside, 1999].

Al-Shalabi and Evens [1998] extended the algorithm of Al-Fedaghi and Al-Anzi [1989] to find the quadrilateral roots for an Arabic word. They enhanced the efficiency of the algorithm by removing the longest possible prefix and looking for the root in the remaining first five characters by comparing patterns with the combination of the first character with two other characters from the second, third and fourth positions. They used the new algorithm to find both triliteral and quadrilateral roots. The algorithm was tested for accuracy and efficiency, but not using IR experiments. It is also not known how the algorithm deals with weak letters [Khoja and Garside, 1999].

Khoja and Garside [1999] introduced a new algorithm that extracts roots from Arabic words. The algorithm is different from the previous morphological analysers in that it uses stopwords and considers weak letters when returning roots. The algorithm uses lists of valid Arabic roots and patterns. After every prefix or suffix removal, the algorithm compares the remaining stem with the patterns. Whenever a pattern matches a stem, the root is extracted and validated against the list of valid roots. If no root is found, the original word is returned untouched. The algorithm is efficient and accurate, but falsely stems proper names and foreign words [Larkey et al., 2007]. It has been evaluated in standard IR experiments and been shown to produce results comparable to light stemming. For example, Larkey et al. [2002] show that mean average precision is improved by 75.77% using the Khoja stemmer.

We use this stemmer to test the effectiveness of root stemming in indexing Arabic text in Chapters 4 and 5. We also test the effects of not stemming foreign words on root stemming in Chapter 6.

Al-Kharashi [1991] and Al-Kharashi and Evens [1994] compared the effectiveness of indexing Arabic text with their Micro-AIRS system using roots, words and stems. Using 355 bibliography records, they manually created a dictionary of 1,126 words, 725 stems and 526

roots which they used to identify roots, words and stems. Using a set of 29 queries and corresponding relevance judgements, they reported that the root-word index outperformed both the stem and the word index, with the word index being the least effective.

Similar experiments were conducted by Abu-Salem [1992] who conducted a series of experiments on using words, stems, and roots as index terms. His experiments on a collection of 120 documents and 32 queries confirmed the conclusions of Al-Kharashi [1991] that root-based indexing outperforms both stem-based and word-based indexing. Abu-Salem used a test collection of 32 queries and a collection of 120 documents. Abu-Salem and Omari used the same system in 1995 to investigate the effects of the inverse-document frequency *idf* weighting function on retrieval performance. These experiments showed that stem-based retrieval is superior to word-based retrieval; they also showed that root-based retrieval is significantly better than word-based retrieval, and significantly better than stem-based retrieval at higher recall levels.

Abu-Salem et al. [1999] tested the effects of three weighting schemes on the performance of the three different retrieval methods. They used the cosine similarity coefficient with a binary weighting scheme, the *tf.idf* weighting scheme, and a mixed stemming method between the query and the document. In the mixed stemming method they used a dictionary of stems, words, and roots along with their respective average weights across all documents to find the best weight for terms in the query. They decide how to index each term based on the best weight of its root, word, or stem. Their results show that the mixed method outperforms the binary weighting method; that the $tf.idf$ weighting scheme with the root and stem indexing methods is superior to other methods; and that the root indexing method is the best of the methods they used.

Hmeidi et al. [1997] compared automatic and manual indexing using words, roots, and stems. They used a test collection of 242 abstracts and 60 queries with relevance judgements, and concluded that automatic indexing performs better than manual indexing when using words as index terms, and when using stems and roots as index terms it is only better than manual indexing at higher recall levels, above 0.3 and 0.5 respectively. Their results show that manual indexing using roots as index terms gives better results than using words and stems. They also concluded that automatic indexing using roots as index terms gives better results than using words.

Finite-State Transducers have been used to analyse morphology of many languages including Arabic [Narayanan and Hashem, 1993]. Morphological analysers use a finite-state transducer to analyse words based on rules that govern the combination of morphemes and

on rules of word structure. A two-level finite-state transducer has been proposed for Arabic by Beesley et al. [1989] in which a lexicon and a set of parallel rules are used. This transducer was implemented by Beesley [1991] for the ALPNET project, and later converted to the Xerox Finite-State Morphology format to overcome limitations such as manual rule compilations and lack of speed [Beesley, 1991]. The new system [Beesley, 1998] uses a root lexicon that includes about 4,930 entries. The system combines these roots with a list of hand-coded patterns to generate stems. It uses a pattern lexicon of about 400 phonologically distinct patterns, and other lexicons of prefixes, suffixes, and non-root-based stems. Using these lexicons, the analyser generates about 72,000,000 words that can be analysed to their possible spellings. Beesley speculates that the system could be improved by adding proper nouns.

According to Darwish and Oard [2002], finite state analysers have been criticised for the excessive manual rule setup, and their restriction to words found in their Arabic dictionaries. They also fail to resolve morphological ambiguity caused by the absence of short vowels in Arabic text [Kiraz, 1998].

Buckwalter [2002] developed an Arabic morphological analyser that returns the possible segmentations of an Arabic word. The analyser uses three lexicons of possible Arabic prefixes, stems and suffixes, and uses three compatibility tables to validate the prefix-stem, stem-suffix, and prefix-suffix combinations. It accepts an Arabic word and provides its possible segmentations — represented using English characters. The underlying lexicons and rules of this system were later updated [Buckwalter, 2004]. The morphological analyser cannot be used directly in IR experiments as it provides more than one possible solution for the same word. Larkey et al. [2007] derived two versions of the analyser and used them in IR experiments using the TREC 2001 and TREC 2002 test collections. In the first version, the analyser is modified to return the normalised stem based on the light10 stemmer normalisation scheme (to be explained in Section 3.1.2). If the analyser fails to analyse the input word or returns more than one distinct stem after normalisation, the normalised version of the input word is used instead. In the second version, such returned input words are stemmed using the light10 stemmer. The analyser performs more poorly than the light stemmer when using the topic titles, but performs comparably when using query expansion.

We modify both versions of Buckwalter analysers to return the first stem of an Arabic word and test their effectiveness and efficiency in stemming Arabic text in Chapters 4, and 5.

Darwish and Oard [2002] developed a morphological analyser called “Sebawai”. The system uses the ALPNET lexicons to estimate the occurrence probabilities of patterns, prefixes,

and suffixes. The aim of this system is to increase coverage by automatically constructing lexicons. The system uses a list of (word, root) pairs that is automatically extracted using the ALPNET morphological analyser. Two lists of Arabic words were passed to the ALPNET analyser, and the successfully analysed pairs — 280,074 in all — were captured. These pairs were then used to estimate the probability of occurrence of prefixes, suffixes, and patterns. The analyser detects roots by analysing a word to determine its possible prefix-stem-suffix structure. It compares the stem with its pattern list and extracts the root which is checked against a list of 10,000 roots to confirm that the root is correct. In case more than one root is determined for an input word, Sebawai ranks results according to estimated probability that a prefix, or a suffix would be observed and that a pattern would be used. Named entities and foreign words cannot be analysed since they do not have roots. The system cannot return one-letter words to their roots, and cannot analyse complex Arabic words that form a complete sentence. Sebawai was successful in analysing 93% of words that ALPNET was able to analyse, and 21% of the words on which ALPNET failed.

Darwish et al. [2005] used both roots and stems returned by Sebawai to index the same collection and compared it with another analyser that considers context [Lee et al., 2003]. The outcome showed that the roots returned by Sebawai lead to lower results than the context-based analyser. Results show that Sebawai's stem-based and root-based indexing methods perform comparably.

Darwish and Oard [2007] showed that indexing the TREC 2001 collection using the roots returned by Sebawai is comparable to word-level indexing, but inferior to indexing stems. They suggest that this divergence from previous published results may be due to the size of the test collection or the insufficient accuracy of the analyser.

Taghva et al. [2005] present the ISRI[11] algorithm that extracts roots similar to the stemmer of Khoja and Garside [1999], but that does not use a root dictionary. This algorithm uses a list of patterns that return three-letter or four-letter roots. These patterns are classified according to their length (4, 5, or 6). It also uses a list of prefixes and suffixes that range in length from 1 to 3. The process of extracting roots starts by removing diacritics and normalising the different forms of *hamza* "ء ، ؤ ، ئ". The algorithm then removes the three-letter and two-letter prefixes in their relevant order, removes the prefix "و", and normalises the different forms of *alef* to a bare *alef* "ا". At this point, words left with three or fewer characters are returned as roots. Words longer than three characters are compared with

[11] Information Science Research Institute.

patterns. A four-letter word is compared with four-character patterns to extract the root. If no match is found, a one-character prefix or suffix is removed and three-character stems is returned as a root. If a word is five characters long, it is compared with five-character patterns that return three-character roots, and then those that return four-character roots. If no match is found, prefixes and suffixes are removed and a root is extracted using four-character patterns. If the remaining stem is six-characters long, the algorithm attempts to extract three-character roots using the corresponding patterns. If no root is extracted, it removes affixes and uses five-character patterns and then four-character patterns to extract the root. Taghva et al. [2005] report that the ISRI approach performed comparably to the Khoja stemmer and to a light stemmer that removes the same affixes but without pattern matching on the TREC 2001 collection.

Attia [2006] stated that both the Xerox Arabic Finite-State Morphology and the Buckwalter Arabic morphological analyser have significant problems in design and coverage. Attia [2006] stated that in both analysers, the inclusion of a large number of classical entries that do not feature in MSA, the use of spelling relaxation rules, the absence of rules that combine words with clitics and affixes, and the use of verb-inflection rules in the passive and the imperative mood contribute to increased ambiguity. Attia attempted to avoid these problems in the course of developing another system using a corpus of 4.5 million words. This system uses the word stem as a base form, and contains 9,741 lemmas, and 2,826 multi-word expression to effectively cover the domain of news articles. The system uses a set of alteration rules to generate the different forms of the word using the stem. The author stated that the system coverage is limited.

Lee et al. [2003] developed an Arabic morphology system that segments words within a sentences to prefix-stem-suffix form. The system adopts a trigram language model and a list of valid prefixes and suffixes. The language model has been estimated from a manually segmented Arabic corpus and re-estimated based on unsupervised acquisition of new stems from a large unsegmented corpus. The system achieved 97% accuracy on a test corpus of 28,440 words. The system does not handle Arabic infixes. Darwish et al. [2005] showed that this system outperforms the Sebawai morphological analyser and the Al-Stem light stemmer (described in 3.1.2) in an IR experiment due to its improved morphological analysis.

Diab et al. [2004] developed a system that uses a support vector machine (SVM) to tokenise words, assigns part-of-speech (POS) tags to words, and annotates phrases in Arabic text. In tokenisation, the system segments clitics (conjunctions, prepositions, and pronouns) from stems; in POS tagging, the system assigns tags to the segmented clitics and

stems with 24 parts-of-speech tags derived from 135 tags used in the Arabic TreeBank POS tagset [Maamouri et al., 2003]; and in the phrase annotation stage, the system chunks Arabic text to non-recursive phrases such as noun, adjectival, and verb phrases. The system has been trained on a sample of the Arabic TreeBank; this contains text from Agence France Presse (AFP) dispatches annotated using the Buckwalter morphological analyser; the annotations are then hand-corrected. The system is reported to achieve 99.77% accuracy on tokenisation and 95.49% accuracy on POS tagging. The system cannot be directly used in IR experiments and needs to be modified to return stems instead of words and tags. Nevertheless, since the text is already annotated, stems can be extracted from it. Larkey et al. [2007] modified this system to generate four different versions and compared their effectiveness in IR using the TREC 2001 and TREC 2002 test collections. They included two modified versions of the Buckwalter analyser and their light10 stemmer. The four versions of the Diab tokeniser perform significantly worse than the light10 stemmer and the two versions of the Buckwalter analyser.

Aragen [Habash, 2004] is a lexeme-based Arabic morphological generator and analyser that uses Buckwalter lexicons and rules in analysing words. However, instead of using a sequence of strings to represent the output, the system uses a set of feature keys mapped to stems, prefixes and suffixes. Feature keys are used to build the feature set in the form of lexeme-and-feature rather than prefix-stem-suffix. Habash and Rambow [2005]; Habash [2007] produced a derivative of this system, called Almorgeana, to annotate Arabic dialects for machine translation applications. With the help of morphological disambiguation, the system is reported to exhibit an accuracy of up to 98.1% in tagging Arabic words correctly using the Arabic TreeBank text.

Morphology aids in distinguishing affixes in Arabic words. Intensive analysis of Arabic words, however, has been shown to be unhelpful for AIR; it also requires comprehensive lists of prefixes, suffixes, stems, roots, and rules to be prepared in advance. Such lists are usually incomplete due to ambiguity and exceptions in the language. For example, broken plural construction has no regular rules, and instead applies patterns. In the absence of diacritics, most analysers would fail to precisely extract roots. We use morphological rules to support stemming in Chapter 4. We use a different approach that relies on terms in existing lexicons or text corpora to predict stems and distinguish affixes from core letters in Arabic words.

3.1.2 Light Stemmers

The idea of Arabic *light-stemming* was initiated by Aljlayl [2002] who implemented a novel light stemmer that aims to remove the most frequent prefixes and suffixes, rather than to find the exact root of an Arabic word.

The stemmer starts by removing diacritics from Arabic words. It then normalises the leading alef with a bare alef. This step is repeated after any prefix removal. The stemmer replaces the final "ى" with "ي"; the sequence "ىء" with "ئ"; the sequence "يء" with "ئ"; and the final "ة" with "ه". It is a requirement that a word has to have three or more characters in order to remove prefixes or suffixes. The first step in removing affixes is to remove the leading conjunction "و", then removing the definite article with any preceding prepositions and conjunctions. The stemmer removes the most common suffixes starting with the longest ones. The stemmer then removes prefixes such as the prepositions "لـ"; and "بـ", and the leading "يـ" if the second character is "تـ". The stemmer uses a list of Arabised — or foreign — words to avoid stemming them. It is not clear when the checking is done. There is no complete list of affixes removed by the algorithm, nor is there any mention of using stopping. The stemmer participated in the TREC 2001 evaluation and was the second-best stemmer out of seven stemmers used in the evaluation. The performance of this stemmer was compared with performance of the Khoja root stemmer, and was reported to add 24.3%, and 19.6% improvement to the root stemmer with and without relevance feedback respectively.

Larkey and Connell [2005] extended the stemmer variants they used in TREC2001 — the light1, light2, light3, and light8 stemmers — to develop their light10 stemmer. All algorithms share the same preliminary normalisation step, where punctuation, diacritics, and non-letters are removed; "آ", "أ", and "إ" are replaced with "ا"; the final "ى" is replaced with "ي"; and the final "ة" is replaced with "ه". The first version, "light1", removes only the definite article with all possible preceding single particles except the preposition "لـ", with the condition that the remaining stem has to have two or more letters. The second version, "light2", removes an additional prefix "و". The third version, "light3", extends light2 to remove the suffixes "ة" and "ه". Further suffixes are removed in the fourth version, "light8". The light10 stemmer comprises light8 with the additional removal of the prefix "للـ" (see Table 3.1). All these stemmers remove suffixes in the same order from right to left as long as the remaining stem has three or more letters. In experiments using the TREC 2001 collection and the first four stemmer variants [Larkey et al., 2002], and using the TREC 2001

Stemmer	Prefixes Removed	Suffixes Removed
Aljalyal	و، ال، وال، كال، ست، سي، ل، ب، ت، ي، للّ، ال	ين، ون، ات، ة، ان، ي، هم، هن
light10	للّ، ال، وال، بال، كال، فال، و	ها، ان، ات، ون، ين، يه، ية، ه، ة، ي
Al-Stem	وال، فال، بال، بت، يت، لت، مت، وت، ست، لم، بم، نت، وم، كم، فم، ال، للّ، وي، لي، في، وا، فا، لا، با	ات، وا، ون، وه، ان، تي، ته، تم ، كم ، هم، هن، ها، ية، تك، نا، ين، يه، ة، ه، ي، ا
Chen	وال، بال، فال، كال، وللّ، مال، اال، سال، لال، فا، كا، ول، وي، وس، سي، لا، وب، وت، وم، للّ، با و، ب، ل	ها، ية، هم، ن، ما، و، يا، ني، يا، ه، كم ، كن، تم ، تن، ين، ان، ات، ون ة، ه، ي، ت

Table 3.1: Prefixes and suffixes removed by the Arabic light stemmers described in Section 3.1.2.

and TREC 2002 collections and all five stemmer variants [Larkey and Connell, 2005; Larkey et al., 2007], each variant was shown to be better than its predecessor, with the exception of light10 compared to light8, where the improvement was not significant. The same collection was stemmed using the Khoja root stemmer. The light10 stemmer significantly outperformed the Khoja stemmer, but the light8 stemmer did not exhibit any significant difference. The stemmer also compared favourably to the Buckwalter analyser and Diab tokeniser, except when the Buckwalter analyser was used with query expansion. The light10 stemmer is publicly available as part of the Lemur Toolkit.[12]

We use the light10 stemmer as a baseline to test improving light stemming using morphological rules to avoid stemming core letters in Arabic words in Chapter 4.

The Al-Stem light stemmer of Darwish and Oard [2003b] removes punctuation and diacritics, with two normalisation options. In the first option, only the different forms of alef are normalised to the bare alef. In the second option, the characters "ؤ","ئ", and "ء" are normalised to "ا". The stemmer removes 24 prefixes and 21 suffixes (see Table 3.1). No stopwords are removed. This stemmer has been compared with the light8 and a modified version of it. In experiments using the TREC 2001 and TREC 2002 collections, the modified stemmer was shown to be significantly better than both light8 and Al-Stem [Darwish and Oard, 2003b].

Al Ameed et al. [2005] have developed five light stemmers to enhance the Al-Stem light stemmer developed by Darwish and Oard [2003b]. They used more affixes and proposed two ways to remove prefixes and suffixes. They evaluated their algorithms using a list of more

[12]http://www.lemurproject.org

than 1,450 words. They measured algorithms by their ability to return meaningful words, and by how frequently affixes were removed. They concluded that their algorithms produced more meaningful results than Al-Stem.

Chen and Gey [2002] described two stemming algorithms. The first is an MT-based stemmer that clusters Arabic words based on their English translation. Arabic words are translated into English using an Arabic-English dictionary, words that map to English stopwords are removed, and Arabic words that translate to the same English word are replaced with the shortest Arabic version. The second is a light stemmer — referred to as Chen — that removes prefixes and suffixes from Arabic words. They derived their list of prefixes and suffixes according to their grammatical functions and their frequency of occurrence in the unique words of the TREC 2001 corpus. In total, the stemmer non-recursively removes 26 prefixes and recursively removes 22 suffixes (see Table 3.1). The stemmer starts by removing the three-letter prefixes if the Arabic word is at least five letters long, then the two-letter prefixes and the "و" prefix if the word is at least four characters long. It removes the prepositions "لـ" and "بـ" only if the word is at least four characters long and the remaining string exists as a separate word in the document collection. Two-character suffixes are then removed recursively. Finally the single-letter suffixes are removed recursively as long as the word is at least three characters long. The stemmer uses a stopword list created by translating the unique words of the TREC 2002 collection to English and then considering those words that translate only to English stopwords to be Arabic stopwords. While the list of English stopwords contains 360 entries, the list of Arabic stopwords derived in this manner contains 3,447 words.

Kadri and Nie [2006] compared linguistic-based stemming with light stemming. For linguistic-based stemming, they used corpus statistics to resolve ambiguity about whether a letter sequence is a proper prefix or suffix. They used the TREC 2001 corpus to construct all possible stems and their frequency of occurrence in the corpus. To stem a word, they decomposed it to its possible stems and selected the most likely candidate based on its statistics in the corpus. In the light stemming approach, they built a stemmer that truncates the most frequent prefixes and suffixes in the same corpus. They constructed a list of 413 Arabic stopwords and normalised the text using a similar approach to Aljlayl [2002]. From a comparison of the two systems using the TREC 2001 and TREC 2002 test collections they concluded that using linguistic-based stemming produces better results than the light stemming, and that the light stemming "is not the best approach for Arabic IR".

3.1.3 Statistical Approaches to Arabic Stemming

Statistical methods have also been used to stem Arabic words. These approaches involve the use of n-grams, where a word is segmented into a number of overlapping equal size text fragments of n characters. Similarity measures are used to group similar words based on the similarity of their n-grams.

AlShehri [2002] studied the statistical characteristics of Arabic words and their overlapping n-grams using six Arabic corpora. He recommended that the optimal n-gram size for indexing and retrieving Arabic text is 3. He compared the effectiveness of using tri-grams and a mix of 3, 4, and 5-grams as index terms to the word indexing approach. He reported experiments using two test collections: one containing 242 Arabic scientific abstracts and 60 queries, and the second containing 187 full newspaper articles from the Al-Raya newspaper and 30 queries. He showed that both n-gram indexing methods significantly outperform the word indexing method on the first collection but not on the second.

Xu et al. [2002] tested using 2, 3 and 4-grams to index words and stems produced by the Buckwalter morphological analyser. The stem-based n-grams generally outperformed the word-based n-grams. When using stem-based indexing, 3-grams outperformed 2-grams and 4-grams by 5%, although this margin was not statistically significant.

From initial experiments using a text collection of 4,000 documents and 25 queries, Darwish et al. [2001] concluded that using different gram sizes from words and roots results in improved retrieval. Indexing grams of size 3 to 5 for words, and of size 2 to 4 for roots, outperforms the root, word and stem indexing, but not the combination of word and root indexing. They showed that indexing text using a combination of words, roots and their possible grams is superior to all indexing techniques involved in the comparison. Using the TREC 2001 test collection, they formulated queries from the title and the description fields and indexed the text as in the initial experiments, but added another index that uses a combination of roots, stems, and words. Their results on monolingual and cross-lingual retrieval show that indexing word trigrams outperforms all other techniques. The authors used the root as the index term in subsequent monolingual experiments and found that the mean average precision is significantly better than other indexing techniques.

Larkey et al. [2002] used a statistical approach to Arabic stemming that does not involve n-grams. Their approach is based on the analysis of the co-occurrence of terms in Arabic text. They first stemmed Arabic text using their light stemmers and the Khoja stemmer. They then removed vowels from the remaining strings to form large classes of words. They

refined these classes by calculating word co-occurrences for all words in every class, and then repartitioned classes according to word co-occurrences. They used a variant measure called *em* "to calculate the proportion of word co-occurrences that are over and above what would be expected by chance" and to repartition word classes. This approach adds a significant improvement over their light stemmer including light2, and light8 stemmers, but not over the Khoja stemmer.

Mustafa and Al-Radaideh [2004] explored searching Arabic text using n-grams. They used bigrams and trigrams to search a set of 6,000 distinct words selected from several text documents. They formed 50 queries and used the Dice similarity measure to find variants in the list. They considered words with a similarity value above 0.6 to be related. They concluded that the use of infixes in Arabic caused word variants to exhibit low similarity using the Dice measure, and recommended against the use of n-grams for Arabic text retrieval.

3.2 Retrieval of Foreign Words

Finding variants of names is a problem that has been long recognised in information retrieval and has been addressed in great depth by the database community [Raghavan and Allan, 2005]. Few studies have tested the retrieval of name variants in the context of IR where names are to be located within text documents rather than from a name databases. In this section, we report experiments conducted to find name variants.

Zobel and Dart [1995] used two lists of English words to test the effectiveness of phonological and string similarity techniques in retrieving wrongly-spelt words and name variants, and evaluated efficiency using another list. The first list contains 113,212 words and 117 misspelled words as queries. Results of these queries are the correct respective words in the list. The second data set contains 31,763 distinct English personal names extracted from student names with 48 randomly-chosen names used as queries. Results are evaluated manually based on the top 200 answers returned by the different techniques. The third set contains 1,073,727 distinct words extracted from the TREC text collection; this set does not contain relevance judgements, and is used to evaluate computation time and space requirements using the 48 names from the second data set as queries. The study compared 9 techniques including Edit Distance, gramCount, gramDist, Soundex, Phonix, and agrep. Their results show that Edit Distance retrieves name variants with a precision of 63.7%, followed by gram-dist (61.5%), gramCount (55.9%), and agrep (32.8%); the phonetic techniques are shown to be the weakest in the experiment.

Zobel and Dart [1996] used a list of over than 30,000 distinct English names extracted from the Web to test the performance of phonetic and string similarity techniques in identifying names. They created 100 queries by randomly selecting surnames from the White Pages telephone directory for Melbourne, Australia, and used pooling to draw up their relevance judgments. To ensure that judgements are based on the similarity of sounds; the judgements were created using two assessors, where one reads aloud each query and its potential match, and the other judges whether they sound similar. Using three sets of judgments, Zobel and Dart compared the performance of nine similarity techniques: Editex, Edit Distance, Ipadist, Tapered Editex, Tapered Edit Distance, Q-grams, Best agrep, Phonix+, Phonix, and Soundex. In order to evaluate such techniques and avoid the problem of weak ordering, they computed average recall and precision over ten random permutations. Their results show that the Editex technique outperforms other techniques. It is followed in turn by the Ipadist, Tapered Editex, Edit Distance and then Q-grams algorithms. The phonetic techniques performed weaker than the baseline — finding strings that are different from the query string at most by one character.

Pfeifer et al. [1995; 1996] created their COMPLETE test collection that contains 14,972 distinct names from different sources and 90 names chosen at random from the collection for use as queries. The relevant names are determined manually for each query. There are a total of 1,187 relevant names for the 90 queries. This test collection is used to test the effectiveness of finding name variants using phonetic and string similarity techniques including Soundex, Phonix, bigrams, and trigrams. They have also modified the Phonix algorithm to encode the first 4 characters (Phonix4), to encode the first 8 characters (Phonix8), and to encode the first 4 characters plus 4-byte ending sound (PhonixE). Their results show that all similarity techniques are significantly better than the exact-match technique, and that the tailed bigrams perform better than other techniques. They also report that the combination of the tailed bigrams and PhonixE is better than the performance of any single technique.

are have differentPirkola et al. [2002] introduced the targeted s-gram technique to find variants of names in English, German and Swedish in a list of 199,000 OOV Finish words. They show that this technique is more effective than conventional n-gram matching in finding similar short names.

Holmes and McCabe [2002] used the COMPLETE test collection of [Pfeifer et al., 1996] to test the effectiveness of the Russell and Celko Soundex algorithms, their own fuzzy Soundex, fusion, and code shifting in finding name variants. Their findings show that Soundex has the worst average precision while the combination of all the techniques produced the best

results, finding 96% of all relevant names with a precision of 70%.

Holmes et al. [2004] used n-grams to enhance finding transliterated Arabic names in English. The algorithm uses 45 transformation rules to normalise the transliterated names, and then generates n-grams from the enhanced versions of the names. Similarity is computed based on the shared n-grams using the Dice coefficient. Results show that this algorithm achieves an average precision of 90% with a recall of 100%. The evaluation is carried out using a collection of 5,819 Arabic first names with 150 queries that have variants in the collection.

Ruibin and Yun [2005] and Gong and Chan [2006] used the COMPLETE test collection and the test collection of Zobel and Dart [1996] to evaluate a new technique based on syllable alignment. The algorithm segments phonetic strings into syllables and compares strings based on syllables rather than letters. The algorithm performs better than the Edit Distance and Editex algorithms using the COMPLETE test collection, but not when using the collection of Zobel and Dart.

Christen [2006a;b] used four name corpora to compare approximate-matching algorithms. Three of these corpora were formed by extracting unique names from healthcare records and generating random new name pairs, while the fourth was the COMPLETE collection of Pfeifer et al. [1996]. After evaluating 24 techniques, Christen concludes that "there is no single best name matching technique" and that techniques should be chosen based on the data in hand.

Aqeel et al. [2006] compared the effectiveness of finding Arabic name variants and misspelled words using two new algorithms and other language-independent similarity techniques such as Edit Distance and n-grams. They formed a test set of 7,939 names along with two sets of queries that were created by altering some of these names by adding, deleting, or inserting characters. The first algorithm is based on the Russell Soundex and the second is based on n-grams. Their phonetic "ASoundex-final" algorithm encodes Arabic characters including long vowels into 11 groups. Unlike the Russell Soundex, this algorithm does not restrict the encoding to four characters. The final version of the algorithm "ASoundex" uses ASoundex-final to generate multiple encoded versions of length 2 to 9, and then employs fusion to generate the best possible result. Their "Tanween-aware n-grams" approach considers only the diacritics used for tanween and shadda in the generation of n-grams. Their results show that the ASoundex algorithm significantly outperforms the n-gram approach and Edit Distance, and that the combination of ASoundex and Edit Distance leads to the best results.

We check the effectiveness of using the ASoundex-final algorithm in grouping variants

of foreign words in Arabic in Chapter 7, as this is the only phonetic algorithm available for Arabic.

The effectiveness of using different spelling name variants to improve document retrieval performance has been explored by the CLIR community. This often involves the use of a bilingual dictionary to translate words and transliterate OOV words in the query to the target language.

Larkey et al. [2003] demonstrated the importance of handling translating proper names in CLIR experiments. They tested the effectiveness of using several translation and transliteration sources in improving retrieval in the context of a CLIR task. They expanded English queries and then translated them into Arabic using different dictionaries. In total they identified 241 proper names in the English queries. Not transliterating names in the queries resulted in performance around 57% lower than when the names were transliterated. Expanding Arabic queries with the top 20 transliterations scores the best average precision. They show that retrieval effectiveness is affected by the quality of the dictionary, and recommend that unknown proper nouns be transliterated for improved effectiveness.

Abduljaleel and Larkey [2002] implemented an n-gram technique to transliterate English words into Arabic. The effectiveness of this technique in an IR context was tested by Abduljaleel and Larkey [2003] and compared to a hand-crafted transliteration model. The task was to use English queries to search an Arabic text collection. To test the effectiveness of transliteration on retrieval performance, they translated queries using the bilingual dictionary of Larkey and Connell [2001] and expanding queries by automatically transliterating all names; only names that are not found in the dictionary; and all unknown words in the query. Only the first twenty transliterations are included. Their results show that expanding queries using different transliterations generally increases the performance over the baseline, and that the hand-crafted model produced a significant improvement in all three cases. The n-gram model results in a significant improvement when transliterating names and words that are not found in the dictionary.

Raghavan and Allan [2004] took a different approach to test approximate string matching to normalise English name variants across all ASR documents by replacing variants with their Soundex code, and then computed the similarity between documents using the cosine similarity measure to determine whether they are related to the same story as part of the Story Link Detection task of the Topic Detection and Tracking (TDT) forum.[13] They used

[13] http://www.nist.gov/speech/tests/tdt

the TDT3 test collection that contains 67,111 broadcast news from Arabic, English and Chinese. For the broadcast news, the ASR output is provided along with its machine-translation version in English. The collection has 60 topics, each with the relevant documents annotated. The similarity between two documents is calculated using the cosine metrics before and after name variant replacement using the Soundex codes. Their results showed a degraded performance due to the poor performance of named entity recognition on the poorly structured ASR text. They tested the same technique on a newswire text collection that contains 4,752 pairs of stories. Using this collection, they achieved 10% improvement suggesting that this technique can improve retrieval.

Raghavan and Allan [2005] tested Edit Distance and four models trained to find name variants using a parallel ASR text and manual transcripts. They formed a baseline by obtaining 296 OOV words and enlisted students to generate 35 groups containing variants of these words. They evaluated all other techniques based on the concept of overstemming and understemming used by Paice [1996] to evaluate stemming algorithms. Their results show that their models are better than Edit Distance in conflating names. They also concluded that using one step as a threshold in the Edit Distance technique to determine similarity between names is better than using two, three, four, or five steps. To test the retrieval of name variants within documents, they used the 35 manually generated name variants, and the TDT3 corpus for testing. They removed any names that did not exist in corpus from the 35 groups, leaving 76 names in total. They considered any document containing at least one of the names or name variants to be relevant. They found that using their algorithms and the Edit Distance algorithm add a significant improvement over the baseline, and that the Edit Distance algorithm produced the best F_1 value. They reported that using their techniques on the TDT3 spoken retrieval task increased MAP significantly over a baseline that used string Edit Distance.

Virga and Khudanpur [2003] tested transliteration to improve retrieval on ASR documents. They indexed words from the TDT2 Chinese collection, and used Mandarin text documents as queries for their baseline approach. Using the character-bigram improves the retrieval significantly. They tested retrieval using English documents as queries. They first translated English documents without transliterating proper names and then included transliterated names, which improved results slightly - albeit not significantly.

A related area of research is personal name resolution, which aims to disambiguate name variants, and to identify other names that are not variants but that refer to a particular individual. In recent work on Arabic, Magdy et al. [2007] use a support vector machine to

classify and normalise personal names across documents. Their classification approach calculates similarity between names using different rules including Edit Distance and a phonetic Edit Distance approach — similar to the English Editex. They evaluated their technique based on purity, entropy and show that their technique produces accurate clusters.

There has been only limited research on conflating variants of names in Arabic languages. This is due to the fact that Arabic names are distinct and have no variants except in writing styles. Most of these errors can be handled by removing diacritics. This can be clearly seen in the work of Aqeel et al. [2006], who generated a data set for their ASoundex algorithm by altering Arabic names and including diacritics. We believe that handling foreign words in Arabic would benefit from such techniques as it is the only category of Arabic words that is characterised by different versions. We test techniques to normalise foreign words in Arabic in Chapter 7.

3.3 Identification of Foreign Words

Identifying names in text has been studied and shown to improve the performance of IR systems. Named Entity Recognition is concerned with identifying names of people, places, and organisations within text. Many systems have been developed to identify named entities within English text, but only a few have been developed for Arabic [Florian et al., 2004; Shaalan and Raza, 2007; Benajiba et al., 2007]. Arabic names rarely have variants, and most variants that do exist typically vary only in diacritics or the letters used, which can be addressed through normalisation. In this thesis, we explore a more challenging problem: how to identify foreign words in Arabic text.

Perhaps the easiest way to identify foreign words is to use dictionaries. Abduljaleel and Larkey [2003] for example, used this method to identify OOV words in English queries and transliterate them into probable Arabic forms. In contrast, we aim to identify foreign words as a broader general class of terms, distinct from Arabic words.

Stalls and Knight [1998] describe research to determine the original word from its Arabic version; this is known as *back transliteration*. However, rather than using automatic methods to identify foreign words, they used a list of 2,800 names to test the accuracy of back transliteration algorithm. Of these, only 900 names were successfully transliterated to their original form. While this approach can be used to identify transliterated foreign words, its effectiveness is not known on normal Arabic words, as only names were used to test the algorithm.

Jeong et al. [1999] used statistical differences in syllable unigram and bigram patterns between pure Korean words and foreign words to identify foreign words in Korean documents. This approach was later enhanced by Kang and Choi [2002] to incorporate word segmentation.

A related area is language identification, where statistics derived from a language model are used to automatically identify languages. Dunning [1994] used n-gram statistics to identify several languages. In their approach, they constructed a language profile by generating overlapping n-grams of text written in each of the language under study. The frequency of every n-gram is calculated and the final language profile is built by ordering its n-grams in order of decreasing frequency. To classify a document language, they generate an n-gram profile for that document in a similar way, and compute the total distance between the n-grams in the document profile and the profile for each language by subtracting the positions of similar n-grams in both lists. The language with the profile closest to that of the document is considered to be the correct language. The authors used the 300 most frequent n-grams to build each language profile, and concluded that this produces good accuracy for strings with fifty or more characters, and works moderately well with strings of ten characters.

Recently Goldberg and Elhadad [2008] have used a statistical model based on a Naïve Bayes classifier to identify foreign words in Hebrew. They used old Hebrew scripts to train their statistical model to learn native Hebrew words, and used an automatic list of transliterated words to train the model to learn the pattern of foreign words. They report a recall of 82% at a precision of 80% in identifying 368 foreign words in a collection of 4,044 words. By combining the statistical model with a Hebrew lexicon, they achieved a recall of 70% with a precision of 91%. Time constraints prevented us from evaluating and applying this recent work for Arabic text.

In Chapter 6, we use the n-grams approach used in language identification to identify foreign words in Arabic text. We also use lexicons, patterns and morphological rules to enhance foreign words identification in Arabic text.

3.4 Chapter Summary

Due to the nature of the language, most published work on Arabic Information Retrieval (AIR) grapples with Arabic morphology. Almost all systems described before the introduction of the Arabic track in TREC 2001 include morphological analysers. The main objective of these systems is to extract the root of an Arabic word. Experiments using small collections have indicated that root indexing is more effective than both stem-indexing and

word-indexing. Light stemmers are a more recent development, and have been shown — on large test collections — to be more effective than root stemming. However, few improvements to stemming approaches have been published in the last five years. Some statistical approaches to AIR have been tested, and trigrams have been reported to be the best gram size for indexing Arabic text. Recent work on AIR systems applies language morphology for part-of-speech tagging for collections in Arabic, and increasingly for the various Arabic dialects.

Many approximate string-matching techniques — including phonological-matching and string-matching approaches — have been developed for English and other European languages; however, these have mostly been tested using a list of names, rather than on a text corpus where other words greatly affect retrieval effectiveness.

The expansion of Out-of-Vocabulary (OOV) words to their variants in a CLIR English-Arabic task has proven that this technique is effective and improves retrieval. Experiments on normalising name variants across ASR documents for retrieval have shown that accurate identification of name variants is critical. Foreign words in Arabic have different variants, identification of such words is crucial to allow unifying them for effective searching; however, there is a dearth of published empirical results on this topic. In Chapter 6, we explore identification of foreign words, and in Chapter 7, we explore techniques to unify their variants.

We continue in the next chapter with a discussion of our work on stemming Arabic words.

Chapter 4

Stemming Arabic

Stemming is the process of merging different forms of the same word that are semantically equivalent and share the same stem [Paice, 1996]. For IR systems, stemming is used to conflate words together in order to increase performance and reduce index space.

Arabic words have many forms. For example, a noun can have up to 519 different forms, while a verb can have up to 2 552 [Attia, 2006]. To convert words to their root or stem, additional letters that attach to the word either at the beginning (prefix), middle (infix), or at the end (suffix) have to be removed by *stemming*. For instance, the words "يكتب" (/jkt̲b/⟨writes⟩), "مكتَبَة" (/mkt̲aba/⟨a library⟩), and "مكتَب" (/mkt̲ab/⟨an office⟩) reduce to "كَتَبَ" (/kat̲aba/⟨wrote⟩) after stemming.

As described in Section 3.1, Arabic stemmers range from deep morphological analysers to light prefix-suffix removers. Stemmers generally remove affixes by comparing the specific parts of the word with a pre-prepared list of affixes [Al-Sughaiyer and Al-Kharashi, 2004]. These lists are usually built based on the language morphology and statistical analysis of Arabic text [Aljlayl and Frieder, 2002]. Using such a fixed list to match the beginning or the end of the word is effective [Larkey et al., 2002; Aljlayl and Frieder, 2002] but also affects core letters. This can happen in any language, but is a major problem for Arabic, where pronouns conjunctions, prepositions, and particles are attached directly to words. The same character sequence may also be core characters, and removing such core characters leads to incorrect results.

In this chapter, we examine approaches for the proper removal of affixes using lexical Arabic grammar rules. We empirically compare approaches to normal affix removal, and show that our technique increases text retrieval effectiveness. We explore using the corpus as

a lexicon, and show that it is possible to satisfactorily stem Arabic without a comprehensive lexicon. We also check whether the techniques developed using normal Arabic text also apply to Arabic text extracted automatically from recorded speech.

4.1 Evaluation of Existing AIR Stemmers

In this section, we compare most of the existing AIR stemmers described in Section 3.1. In order to evaluate existing stemmers, we implemented some of them and modified some others to be used directly in IR experiments.

4.1.1 Stemmers

We used the following stemmers:

Khoja: our implementation of the Khoja stemmer that supports stemming text in large document collections.

B.Stem: Buckwalter 1.0 stemmer [Buckwalter, 2002], modified to return only the first returned stem for each given word.

B.Stem2: As above, for the Buckwalter 2.0 stemmer [Buckwalter, 2004].

B.Lemma: Buckwalter 1.0 stemmer modified to return the lemma for a given word.

light10: Larkey light10 stemmer [Larkey et al., 2007], which is part of the Lemur toolkit.[1]

Al-Stem: Al-Stem stemmer developed by [Darwish and Oard, 2003a].

Al-StemN: As above, but omitting numbers.

noStemming: Removing diacritics and punctuation.

All stemmers, except for Al-Stem, are modified to remove the same stopwords removed by the Khoja stemmer. This has been used by the light10 stemmer and it is available with the Lemur toolkit.

4.1.2 Other Experimental Settings

For the baseline, we used the original 25 TREC 2001 and 50 TREC 2002 queries in a single 75-query set, following the practice of Larkey et al. [2007] and Darwish and Oard [2003b] in

[1] http://www.lemurproject.org

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
noStemming	0.188	0.440	0.430	0.200	0.284	0.650	0.196	0.336	0.559
light10	**0.391**	0.576	**0.674**	**0.291**	0.388	**0.758**	**0.324**	**0.451**	**0.724**
B.Lemma	0.333	0.572	0.561	0.282	0.374	0.716	0.299	0.440	0.652
B.Stem	0.357	**0.592**	0.614	0.280	0.380	0.706	0.306	**0.451**	0.668
B.Stem2	0.311	0.528	0.609	0.284	**0.396**	0.731	0.293	0.440	0.681
Al-Stem	0.362	0.560	0.606	0.251	0.352	0.674	0.288	0.421	0.646
Al-StemN	0.371	0.564	0.628	0.254	0.368	0.695	0.293	0.433	0.668
Khoja	0.264	0.472	0.555	0.237	0.332	0.671	0.246	0.379	0.623

Table 4.1: Performance of existing Arabic stemmers on the TREC 2001 and TREC 2002 collections. All stemmers add significant improvement over the noStemming approach. The light10 stemmer is the best performer, while the Khoja stemmer is the worst.

combining the queries across the two sets. All results in this chapter are drawn up based on the combined set. We use the short queries only represented in the title field in the query set. This has been decided as to imitate the real web search carried out by users as less than 4% of queries submitted by typical internet users have more than 6 terms [Jansen et al., 1998].

We use the Lemur toolkit (described in Section 2.3.6) to run all IR experiments as it supports indexing Arabic text documents. We set the retrieval parameters to use the Okapi BM25 weighting scheme with default values determined for English (k_1= 1.2, k_3= 7, and b= 0.75) (refer to Section 2.2.3). To investigate the effectiveness of relevance feedback, we set the Lemur toolkit to use the top 20 terms from the first 15 returned documents. This was set based on the conclusions reached by Aljlayl [2002] using the TREC 2001 test collection (refer to Section 2.2.4 for more details).

4.1.3 Results

We show results in Table 4.1 and Figure 4.1. All stemmers add a significant improvement in the mean average precision over the noStemming approach. All Stemmers, except Khoja, add significant improvement in all measures [t-test, $p < 0.001$]. The Khoja stemmer adds a significant improvement in only MAP [t-test, $p = 0.019$].

The light10 stemmer MAP is significantly better than the Khoja stemmer [t-test, $p < 0.001$], Al-Stem [t-test, $p = 0.001$], Al-StemN [t-test, $p = 0.003$], B.Stem2 [t-test, $p = 0.013$],

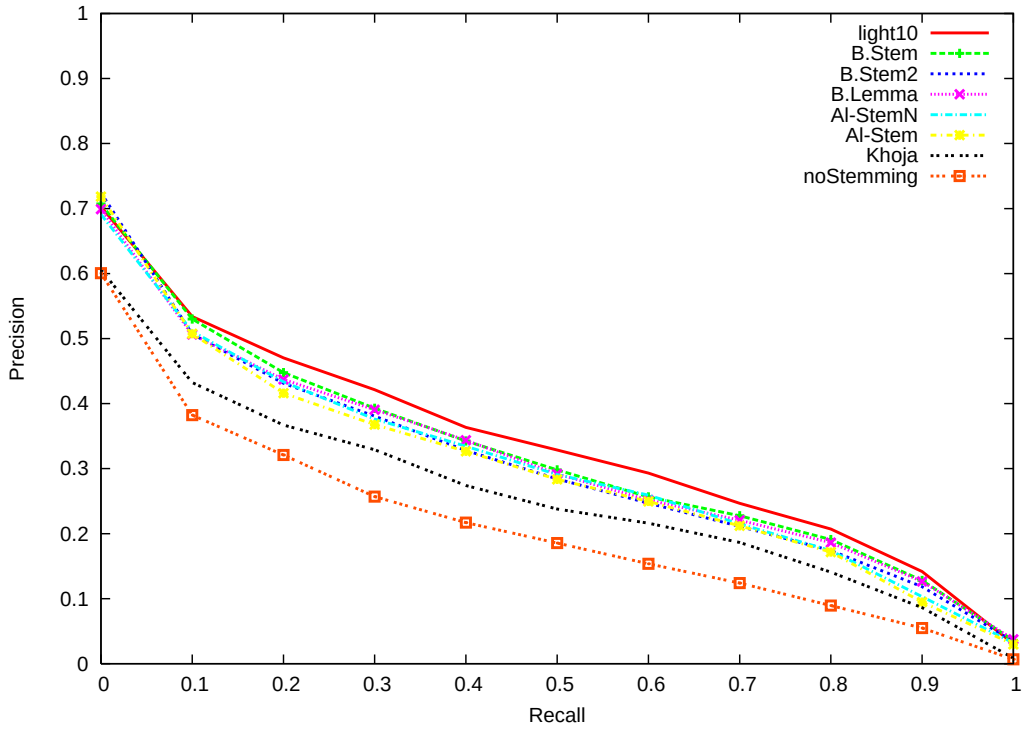

Figure 4.1: Performance of the existing AIR stemmers using the TREC 2001 and TREC 2002 test collections.

B.Lemma [t-test, $p = 0.021$], and B.Stem [t-test, $p = 0.053$]. Although light10 shows a better P@10 value than other stemmers, it is significantly better than only the noStemming and the Khoja stemmer. The stemmer has the highest recall, but it is not significantly better than the Buckwalter stemmers or the Khoja stemmer.

The Buckwalter stemmer, B.Stem, is significantly better than only the Khoja stemmer [t-test, $p = 0.001$] and the noStemming approach.

As described in Section 2.2.4, automatic query expansion and pseudo relevance feedback have been shown to improve Arabic information retrieval [Larkey et al., 2007; Aljlayl, 2002; Darwish et al., 2005]. In our experiments, we also use pseudo relevance feedback using the top 20 terms from the top 15 retrieved documents. The effects of relevance feedback are shown in Table 4.2 and Figure 4.2. The relevance feedback affects the Buckwalter stemmers B.Stem and B.Stem2 the most. The effectiveness of both stemmers is increased by over 24%, while the effectiveness of B.Lemma, Khoja, Al-Stem, and Al-StemN is increased by over 21%.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
noStemming	0.272	0.504	0.499	0.269	0.338	0.773	0.270	0.393	0.660
light10	0.416	0.644	0.641	0.350	**0.438**	**0.838**	0.372	0.507	0.757
B.Lemma	0.403	0.636	0.677	0.344	0.404	0.837	0.364	0.481	0.771
B.Stem	**0.440**	**0.668**	0.708	**0.351**	0.430	0.836	**0.380**	**0.509**	**0.783**
B.Stem2	0.400	0.620	**0.709**	0.348	0.428	0.834	0.365	0.492	**0.783**
Al-Stem	0.391	0.592	0.582	0.329	0.380	0.798	0.350	0.451	0.709
Al-StemN	0.399	0.608	0.583	0.336	0.398	0.809	0.357	0.468	0.716
Khoja	0.273	0.504	0.480	0.314	0.398	0.809	0.300	0.433	0.674

Table 4.2: Performance of existing Arabic stemmers on the TREC 2001 and TREC 2002 collections using relevance feedback. Relevance feedback aids morphological stemmers more than light stemmers.

The light10 effectiveness is increased by only 14%, while the baseline is improved by over 37%.

The B.Stem stemmer produces the best results. It significantly outperforms the Khoja, Al-Stem, Al-StemN, and noStemming approaches, but not the light10 stemmer.

In general, relevance feedback improves results significantly. For example, relevance feedback improves the effectiveness of the light10 stemmer significantly in MAP [t-test, $p < 0.001$], P@10 [t-test, $p = 0.004$], and recall [t-test, $p = 0.006$]. The one exception is seen for the Khoja stemmer, which is not significantly better than the improved baseline results for relevance feedback.

4.1.4 Discussion

It is clear that the light10 stemmer outperforms other stemmers when no expansion is performed. Similar results have also been reported elsewhere [Larkey et al., 2007]. In contrast, the Buckwalter morphological analyser is the best when expansion is performed. Light stemming aggressively removes affixes without validation, while the morphological analysers assure that the removed affixes are valid.

The light10 stemmer is almost 4.25 times faster than B.Stem in stemming the TREC 2001 collection. However, B.Stem has an advantage in saving about 10MB of disk space compared to the light10 stemmer (index size 476MB versus 486MB). In the following sections, we test

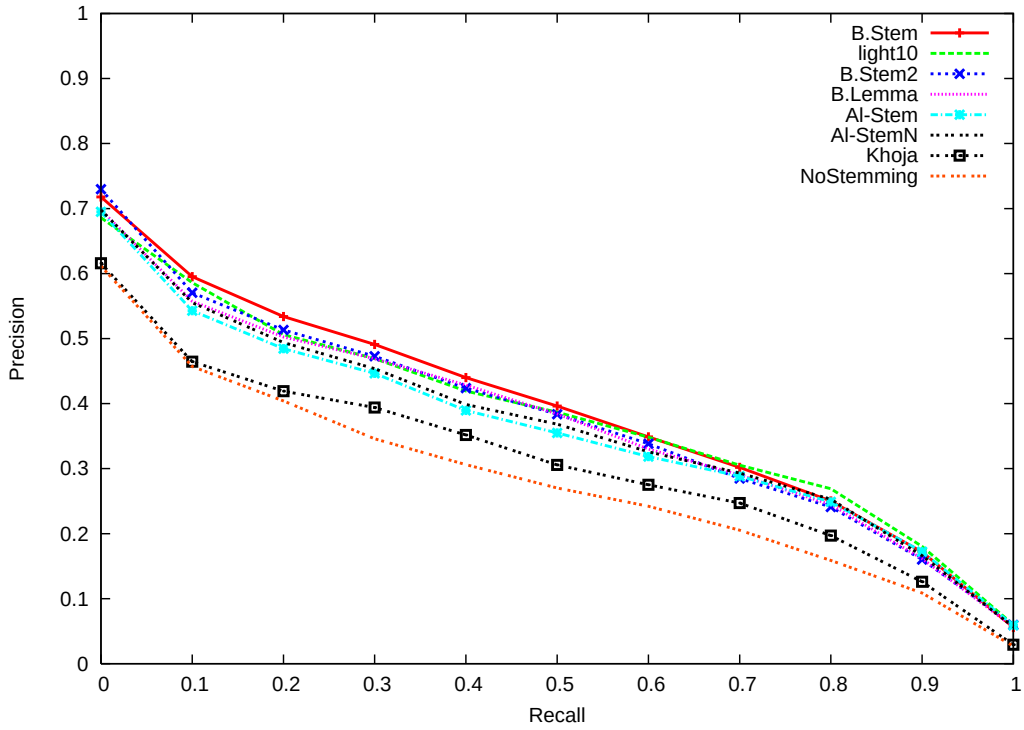

Figure 4.2: Performance of the existing AIR stemmers using the TREC 2001 and TREC 2002 test collections using relevance feedback.

an approach that combines these two approaches and maintains or improves effectiveness and efficiency. We use morphological rules that assure removing affixes, while maintaining or improving effectiveness and efficiency.

4.2 Improving Light Stemming

In this section, we test improving the light stemming using new techniques and supporting affix-removal using morphological rules.

4.2.1 The Baseline

We use the Larkey light10 stemmer as the underlying framework to evaluate the effectiveness of new stemming techniques. We choose the light10 stemmer as it is the best light stemmer publicly available.

To perform stemming that requires lexical validation, we use an Arabic lexicon to validate affixes. We use the Microsoft Office 2003 lexicon [Microsoft Corporation, 2002], which is designed for validating words in Microsoft Office documents. Before removing affixes, we verify that the suspected affixes are not in fact core letters by checking the possible alternative valid morphological forms of the word in our lexicon.

For all techniques used in the light10 stemmer, we test the effects of having that technique in the light10 stemmer, the effects of not using that technique, and the effects of replacing that technique with our new one. Techniques and rules that improve the stemmer or have no negative effects on it are integrated in the final versions of our stemmers. As a convention in this chapter, we show results without relevance feedback in the top part of the result tables, and show results with relevance feedback in the bottom part.

4.2.2 Arabic Text Normalisation

Most AIR stemmers pre-process or normalise Arabic text to unify the different styles of writing Arabic text. A detailed review of many approaches are explained in Section 2.2.1. The first step in the normalisation process is to remove diacritics, punctuation, and other non-Arabic letters. The next step is to normalise the different typographical errors in Arabic writing. To achieve this, we process Arabic text before and after stemming as described in the next subsections.

Arabic Text Pre-processing

Arabic text exhibits different styles of writing, and common mistakes (presented in Section 2.2.1). We have identified several additional variations:

- The combination of both waw "و", and hamza "ء" is written differently by different writers. For example, in the TREC 2001 collection, 88% of the variants of the word "المسؤل" (/alms?ul/⟨the one responsible⟩) are written with the diacritic hamza above the letter waw, as "ؤ", and 12% of cases appear with the diacritic hamza as a separate character after the letter waw, as "وء" in "المسوءل".

- The combination of "ى" and hamza "ء" is written differently by different writers. In some words, they are written as one letter "ئ", and in others as two separate letters "ىء".

- The letter alef "ا" is repeated at the beginning of the word. In some cases it is repeated more than twice which contradicts Arabic writing and morphological rules. We replace two or more consecutive "ا" letters with one letter "ا".

- The letter "ة" can appear only at the end of a word. If the space between a word ending with this letter and the next word is omitted — often deliberately, as the letter "ة" does not affect the following letter — it appears that the letter "ة" is mid-word. For instance the string "قَنَاةُالجَزِيرَة" (/qanaaṯualʒazjra/⟨Al Jazeera channel⟩) is in fact a compound of two words, the first word terminating at the letter "ة". Similarly, the letters, "ر", "و","ز", "ى", "د", and "ذ" frequently adjoin the following word without any space. For example "المُدِيرالعَام" (/almuḍijrulʕaam/⟨the general administrator⟩) is correctly two words, "المدير␣العام".[3] Human readers can generally distinguish such words without problems, but automated stemmers must be adapted to recognise these strings. To correct such mistakes, we propose three techniques:

  - Split any string that has one of the above combinations whenever that combination occurs anywhere in the string after the fourth character, leaving at least three characters in the second string. We call this technique *NormSplit*.
  - Split any string that has one of the above combinations whenever the combination occurs anywhere in the string after the fourth character, leaving two correct strings with at least three characters. We name this technique *CorrectSplit*.
  - Split any string that has the character "ة" or the character "ى" anywhere in the word as these two characters do not appear in the middle of Arabic words. We name this technique *SureSplit*.

Compound Words

In Arabic, some compound names have a distinct form, and are typically written as one word, albeit one that does not comply with grammar rules. For example, the proper noun "بن␣لادِن" (/bnlaḍin/⟨Bin Ladin⟩) is a compound name that has two words which are usually written separately. If the space between words were omitted, the proper noun would become "بنلادن", which has a substantially different appearance. Consequently, the space is never omitted. In contrast, compound names such as "عَبدُ␣الله" (/ʕabḍuallah/⟨Abdu Allah⟩) and

[3] ␣ – represents a blank space

"ابوـعَلِي" (/ʔbwʕalij/⟨Abu Ali⟩) are sometimes written attached "عبدالله" and "ابوعلي" respectively, as the letters of the two words do not change shape when connected. Such variants exist frequently in Arabic, and we must cater for them for satisfactory retrieval performance. For example, the word "عبد" is among the top 100 most frequent words in the TREC 2001 corpus. Leaving such compound nouns unattached when indexing results in the second part — which is usually one of the descriptive names of Allah ⟨God⟩ — being processed as a separate word. For example, the proper noun "عَبدُـالسَّلام" /ʕabdussalam/ meaning ⟨worshipper of the peaceful⟩ becomes "عبد" ⟨worshipper⟩ and "السّلام" ⟨the peaceful⟩, the latter word would then be stemmed to "سلام" (/slaam/⟨peace⟩), or to the root "سَلَمَ" (/salama/⟨survived⟩). This would create more than one reference to such proper nouns.

Before we start stemming we deal with such cases by replacing "عبدـال" with "عبدال" and "ابوـ" with "ابو". This allows us to form a single reference to such proper nouns.

Arabic Text Post-processing

Stemmed words may end with the wrong letter form. For example, stemming the suffix "ـها" (/ha/⟨the feminine pronoun "her"⟩) from the word "مَدرَسَتُها" (/madrasatuha/⟨her school⟩) leaves the word "مدرستـ", which is not a correct Arabic word. Although correctness is not an objective of stemming, leaving such words without any normalisation creates new index terms, and will not group similar words together. In our example, the stem result "مدرست" should be recoded to "مدرسة", or further stemmed to "مدرس". Other instances that need to be normalised include stemming results that end in the letters "ئ" or "ؤ"; these may need to be replaced with a hamza "ء". An example is stemming the suffix "ية" from the word "فضائية" (/fadˤaa?ija/⟨space -adjective-⟩) to result in "فضائ"; this word should be recoded to "فضاء" (/fadˤaa?/⟨space⟩).

Overall Normalisation Approach

As our proposed techniques are based on word validation when removing affixes, we defer some steps of normalisation until the conclusion of affix removal. Before stemming affixes, we

- remove punctuation,
- split words using the SureSplit technique, and
- join compound nouns.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	0.576	0.674	0.291	**0.388**	**0.758**	0.324	0.451	0.724
NormSplit	**0.395**	0.568	0.682	0.291	0.382	0.757	**0.325**	0.444	**0.727**
SureSplit	**0.395**	0.568	**0.683**	0.291	0.380	0.757	**0.325**	0.443	**0.727**
CorrectSplit	**0.395**	0.568	**0.683**	0.291	0.380	0.757	**0.325**	0.443	**0.727**
CompoundS	**0.395**	0.572	0.682	**0.295**	0.382	0.757	**0.325**	0.445	**0.727**
CompoundJ	**0.395**	0.568	0.682	**0.295**	0.386	0.757	**0.325**	0.447	**0.727**
light10Nor	0.376	**0.584**	0.640	0.289	**0.388**	0.754	0.318	**0.453**	0.707↓
light10	0.416	0.644	0.641	**0.350**	0.438	**0.838**	0.372	0.507	0.757
SureSplit	0.426	0.640	0.658	0.346	**0.442**	0.835	0.372	**0.508**	0.763
NormSplit	0.425	0.640	0.658	0.345	0.440	0.835	0.372	0.507	0.762
CorrectSplit	0.425	0.640	0.658	0.345	**0.442**	0.835	0.372	**0.508**	0.762
CompoundS	0.426	0.648	0.677	0.345	0.436	0.835	0.372	0.507	0.770
CompoundJ	0.425	0.640	0.658	0.345	0.440	0.835	0.372	0.507	0.762
light10Nor	**0.442**	**0.688**	**0.697**	0.349	0.428	0.835	**0.377**	0.497	**0.776**

Table 4.3: Effects of normalisation techniques used with light10; the top part of the table shows results without relevance feedback, while the bottom part shows results with relevance feedback. While our individual normalisation techniques add consistent slight improvement with and without relevance feedback, combining them has a negative effect when relevance feedback is not used. ↓ represents significantly different results at the 95% confidence level than the light10 performance.

While parsing, we split incorrect strings using CorrectSplit, and while stemming suffixes, we check the final letter of the word and

- replace a final "ؤ" with "ء" if doing so results in a correct word,
- replace a final "ئ" with "ء" if doing so results, and in a correct word.
- replace a final "ـت" with "ـة" if doing so results in a correct word, and if removing this letter leaves an incorrect word.

After stemming a word, we

- replace "أ", "إ", or "آ" with "ا",
- replace "ى" with "ي",

- replace "ـة " with "ـه",
- replace "وء" with "ؤ",
- replace "یء" with "ئ", and
- replace "اا" with "ا".

Table 4.3shows the effects of adding the individual techniques discussed in this section to the light10 stemmer. CompoundS refers to splitting compound proper nouns, CompoundJ refers to joining compound proper nouns, and lightNor refers to integrating our combined normalisation technique in the light10 stemmer.

Results show that normalising typographical errors improves results. All three techniques slightly improve recall. Joining compound nouns is better than splitting joined ones, and also better than not performing any processing on compound names. The overall recall increases from 0.757 to 0.770 when joining such words and MAP improves slightly, while P@10 decreases slightly.

Our combined approach affects the performance of the light10 stemmer negatively when used without relevance feedback, significantly decreasing recall [t-test, $p = 0.004$], and precision [t-test, $p = 0.073$] but slightly improving P@10. However, with relevance feedback, our combined normalisation technique exhibits better MAP and recall than the light10 stemmer, but lower P@10.

4.2.3 Removing Highly Frequent Words

Stopping is the process of removing highly frequent words in the text in order to increase retrieval effectiveness and reduce index size. We constructed a list of stopwords based on the language structure and characteristics. Our list differs from previously proposed lists (described in Section 2.2.1) in that stopword variants are generated algorithmically based on our classification; this assures that most versions of the stopword are catered for.

We used the El-Khair corpus-based and the light10 stopword lists described in Section 2.2.1, and built a third stopword list by manually extracting well-known pronouns, prepositions and function words from the top 100 most frequent words in the TREC 2001 collection. We normalised words using the light10 normalisation algorithm and kept the unique words after normalisation in each list. We removed variants of the same stopword, keeping only surface forms and classifying words into three different categories: words that can be inflected using suffixes "ه", "هم", "ك", and "ها"; words that can be inflected with

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	**0.391**	0.576	**0.674**	**0.291**	**0.388**	**0.758**	**0.324**	0.451	**0.724**
light10ESW	**0.391**	0.568	**0.674**	0.280	0.384	0.717	0.317	0.445	0.700
light10NSWR	0.230	0.472	0.570	0.275	0.362	0.751	0.284	0.399	0.676
El-KhairSW	0.380	0.592	0.666	0.289	0.382	0.757	0.319	**0.452**	0.719
El-KhairESW	0.286	0.520	0.588	0.234	0.372	0.685	0.251	0.421	0.645
top100SW	0.380	**0.596**	0.669	0.289	0.378	0.756	0.320	0.451	0.720
top100ESW	0.382	**0.596**	0.670	0.278	0.376	0.716	0.313	0.449	0.697
light10	**0.416**	0.644	**0.641**	**0.350**	**0.438**	**0.838**	**0.372**	**0.507**	**0.757**
light10ESW	0.413	0.644	0.633	0.333	0.414	0.796	0.360	0.491	0.729
light10NSWR	0.369	0.624	0.591	0.270	0.362	0.686	0.303	0.449	0.647
El-KhairSW	0.410	0.636	0.631	0.346	0.414	0.833	0.367	0.488	0.750
El-KhairESW	0.411	0.628	0.634	0.337	0.416	0.806	0.362	0.487	0.735
top100SW	0.408	**0.648**	0.617	0.334	0.410	0.818	0.358	0.489	0.735
top100ESW	0.411	0.628	0.630	0.338	0.416	0.818	0.363	0.487	0.741

Table 4.4: Effects of stopword removal on retrieval effectiveness; automatic expansion of stopwords decreases precision and recall of the baseline. The top part of the table shows results without relevance feedback, while the lower part shows results with relevance feedback.

suffixes "ت", "وا", and "ا"; and words that do not accept any suffixes. We also add prefixes "و" and "ف" to every word in the list after adding suffixes. For example, the word "في" (/fj/⟨in⟩) is in the first category that inflects to 18 variants including "فيه" (/fjhi/⟨in him, or in it⟩), "فيها" (/fjha/⟨in her, or in it⟩), "فيهم" (/fjhim/⟨in them⟩), "فيك" (/fjika/⟨in you⟩), and "وفي" (/wafj/⟨and in⟩). From the remaining 109 stopwords in the light10 stoplist, our automatically expanded list produced 672 stopword variants. For each stopword list, we used the original list and the expanded list with the light10 stemmer. Table 4.4 shows the effects of using these three lists and their expanded versions. El-KhairSW and El-KhairESW refer to the light10 stemmer with the El-Khair corpus-based list and the light10 stemmer with the expanded corpus-based list respectively, top100SW refers to the extracted stopwords list from the top 100 most frequent words in the collection, while top100ESW refers to the expanded version of top100SW list, light10ESW refers to the light10 stemmer with the expanded list, and light10NSWR refers to not using any stopwords list with the light10 stemmer.

None of these lists improve light10 performance. In fact they decrease performance. Our attempt to improve stopword removal in Arabic was not successful. Based on these results, we decided to use the default light10 stopword list in our later experiments.

4.2.4 Stemming Conjunctions and Prepositions

Conjunctions and prepositions are widely used in Arabic text, and are difficult to remove without affecting the text. In this section, we classify current approaches to remove particles and then propose three techniques to remove them safely without affecting the core letters of Arabic words in the text.

Classification of Current Particle Removal Approaches

As described in Section 3.1, current AIR stemmers remove particles, but none can remove all particles. Some particles, such as "و" *waw*, are removed by all existing stemmers; other particles, such as "ك" *kaf*, have never been considered on their own in existing stemming approaches. We classify the manner that existing stemmers deal with particles into three general categories:

- Matching the first letter with a pre-prepared list of particles. If a match is found, the first letter is removed as long as the remaining word consists of three or more letters. This approach is used by most current stemmers to deal with a small subset of particles [Chen and Gey, 2002; Darwish and Oard, 2002; Khoja and Garside, 1999; Larkey et al., 2002]. We call this approach *Match and Truncate* (MT).

- Matching the first letter with a list of particles. If a particle is found, the remainder of the word is checked against the list of all words that occur in the document collection. If the stemmed word occurs in the collection, the first letter is considered a particle and removed. This approach was used by Chen and Gey [2002] in conjunction with the other two approaches. We call this approach *Remove and Check* (RC).

- Removing particles with other letters. Certain combinations are generally removed whenever they occur at the beginning of any word. For example, removing a combination of particles and the definite article "ال" ⟨the⟩, particularly, "وال" *wal*, "فال" *fal*, "بال" *bal*, and "كال" *kal*. this approach is used by most current stemmers. We call this approach *Remove With Other Letters* (RW).

Existing stemmers often use a combination of these approaches. They usually start by using the third technique, then continue by removing other particles, particularly "و" *waw* and "ل" *lam*.

Evaluation of Particle Removal Approaches

To check the effectiveness of current approaches for particle removal, we extracted all correct words that start with a possible particle from the TREC 2001 collection.

Words that start with a possible particle constitute 24.4% of this collection. To ensure that we extract only correct words, we check them using the Microsoft Office 2003 Arabic lexicon, and then remove stopwords such as pronouns and separable articles. This procedure results in a list of 152 549 unique correct words that start with a possible particle.

We use three measures to evaluate the effectiveness of the particle removal approaches:

- The number of incorrect words produced; although correct words are not the main target of stemming, an incorrect stem can have a completely different meaning and corresponds to a wrong index cluster. This is particularly true when a core letter is removed from an Arabic word.

- The number of words that remain with an initial letter that could be a particle. This indicates how many possible particles remain after an approach is applied. In Arabic, the second character may be a particle if the first character is a conjunction.

- The number of words actually changed; this measures the *strength* of each approach by counting the number of words that change during stemming [Frakes and Fox, 2003].

Using the assumption that a correct Arabic word with a particle should also be correct without that particle, we experimentally applied the MT, RC, and RW approaches to every word in our collection of unique correct words. The results are shown in Table 4.5.

It can be seen that the MT approach produces a large number of incorrect words (3.39% of all correct words). The results also show that when the MT approach truncates the first letter as a particle, there is a chance that the second letter is also a particle. The proportion of words that still start with letters that could be particles constitutes 14.39% of the total number of correct words. Manual examination of the stemmed list showed that many words have another particle that should be removed, and that many words have their first letter removed despite this letter not being a particle.

Approach	Incorrect words	Possible particles	Altered words
MT	5 164	21 945	151 040
RC	220	41 599	133 163
RW	724	122 878	33 847

Table 4.5: Results of removing particles using current approaches. The number of correct words increased as a result of removing particles indicating the introduction of new incorrect terms in the collection. The MT approach results in the largest number of incorrect words.

The RC approach produces fewer incorrect words. This is because no prefix removal is carried out when the remaining word is not found in the collection. The incorrect words we obtain are due to the collection itself containing many incorrect words. Approximately twice as many words still start with possible particles with RC than MT. This implies that the RC approach leaves the first letter of many words unchanged. This might be desirable, since these might be valid words that do not actually start with a particle. Indeed, manual examination of the result list revealed that many words with particles have been recognised, and particles have been removed correctly. However, the result list also contained a large proportion of words that still start with particles as their first letter.

The RW approach produces a smaller number of incorrect words than the first approach, but generates a very large number of words still starting with possible particles (80.55% of the list of correct words). Moreover, many words are left entirely unchanged.

To conclude, the first approach is too aggressive. It affects Arabic words by removing their first letter, regardless of whether this letter is actually a particle. The second approach, while better at recognising particles in the text, leaves a considerable proportion of words with real particles untouched. More importantly, in many cases one word is modified to another with completely different meaning. The third approach leaves many words unchanged, and deals with only a small subset of particles in the text. It also affects words that start with a combination of particles and other letters, especially proper nouns and foreign words such as "فالوجة" (/falwʒa/⟨the Iraqi city of Fallujah⟩) and "بالتيمور" (/balt̪jmwr/⟨the US city of Baltimore⟩). It is also very hard to recognise such combinations if they are preceded by another particle (conjunction) such as "وبالارض" (/wbalʔrd̪ˤ/⟨and by the land⟩).

New Approaches to Particle Removal

Given the incomplete way in which particles have been dealt with in previous approaches, we have investigated techniques to identify and remove inseparable conjunctions and prepositions from core words in a principled manner. Our methods are based on removing particles using grammatical rules, aiming to decrease the number of incorrect words that are produced by the stemming process, and increasing the completeness of the process by reducing the number of words that still start with a particle after stemming.

We introduce four rules, based on consideration of Arabic grammar, to identify particles in Arabic text. Let L be an Arabic lexicon, P be the set of prepositions {"كـ", "بـ", "لـ"}, C be the set of two conjunctions {"و", "فـ"}, c be a letter in C, p be a letter in P, and w be any word in L. Then:

- Rule 1: Based on grammatical rules of the Arabic language, a correct Arabic word that is prefixed by a particle is also a correct word after that particle is removed. More formally:

$$\forall(p + w) \in L \Rightarrow w \in L \tag{4.1}$$

  and

$$\forall(c + w) \in L \Rightarrow w \in L$$

- Rule 2: Any correct Arabic word should be correct if prefixed by either of the conjunctions "و" or "فـ":

$$\forall w \in L \Rightarrow (c + w) \in L \tag{4.2}$$

- Rule 3: Based on the above two rules, any correct word with a particle prefix should be correct if we replace that prefix with "و" or "فـ":

$$\forall(p + w) \in L \Rightarrow (c + w) \in L \tag{4.3}$$

- Rule 4: Any correct Arabic word that is prefixed by a particle should not be correct if prefixed by the same particle twice, except the particle "لـ" *lam* which can occur twice at the beginning of the word. Let p_1 and p_2 be two particles in $(P \cup C)$, and $p_1 = p_2 \neq lam$, then

$$\forall(p_1 + w) \in L \Rightarrow (p_2 + p_1 + w) \notin L \tag{4.4}$$

Based on these rules, we define three new algorithms: *Remove and Check in Lexicon* (RCL); *Replace and Remove* (RR); and *Replicate and Remove* (RPR).

Stemming the Particle "ـل" *lam*. Due to the peculiarities of the letter "ـل" *lam*, we deal with this letter as a common first step before applying any of our algorithms. Removing the particle "ـل" *lam* from words that start with the combination "ـلل" results in some incorrect words. We therefore deal with this prefix before we deal with the particle "ـل" *lam* by itself. The prefix "ـلل" is a result of adding the particle "ـل" *lam* to one of the following:

- A noun that starts with the definite article. When the particle "ـل" is added to a word with the definite article "ـال" as the first two letters, the first letter "ا" is usually replaced with the letter "ـل" *lam*. For example, "الجَامِعَة" (/alʒaamiʕa/⟨the university⟩) becomes "للجَامِعَة" (/lilʒaamiʕa/⟨for the university⟩). However, if the letter following the definite article is also the letter "ـل" *lam* — as in "اللّقب" ⟨the surname⟩ — the next case applies.

- A noun that starts with the letter "ـل" *lam*. For example, "لَقَب" (/laqab/⟨surname, championship⟩) becomes "لِلَّقَب" (/lillaqab/⟨to the surname⟩) when prefixed by the particle "ـل" *lam*.

- A verb that starts with the letter "ـل" *lam*. For example, لَفّ (/laff/⟨wrapped⟩) becomes "لِلَفّ" (/lilaff/⟨to wrap⟩) when prefixed by the particle "ـل" *lam*.

To stem this combination, we first check whether removing the prefix "ـلل" produces a correct word. If so, we remove the prefix; if not, we try adding the letter "ا" before this word. If the new word is correct, we drop one "ـل" *lam* from the original word.

To remove the particle "ـل" *lam* from words that originally start with the definite article, we replace the first "ـل" *lam* with the letter "ا" and check whether the word exists in the lexicon. If so, we can stem the prefix "ـلل" without needing to check whether the remaining part is correct. If not, we remove the first letter and check whether we can drop the first "ـل" *lam*. This algorithm is used before we start dealing with any other particles in the three following algorithms.

Remove and Check in Lexicon (RCL). In our first algorithm we start by checking the first letter of the word. If it is a possible particle — that is, it is a member of the set P or C — we remove it and check the remaining word in our lexicon. If the remainder is a valid word, the first letter is considered to be a particle, and is removed. Otherwise, the original word is returned unchanged. For example, consider the word "وريد" (/wrjḍ/⟨a blood vein⟩). It starts with the letter "و" as a core letter. Removing this letter from the word

leaves the string "ريد". If the underlying used lexicon contains this word, the first letter is considered as a conjunction and the latter string is returned. This approach differs from the RC approach in that we check the remaining word against a lexicon, rather than against all words occurring in the collection. We expect that this will allow us to better avoid invalid words.

Replace and Remove (RR). Our second algorithm is based on Rule 4.3. If the first letter of the word is a possible particle, we first test whether the remaining string appears in our lexicon. If it does, we replace the first letter of the original string with "و" *waw* and "ف" *faa* in turn, and test whether the new string is also a valid word. If both of the new instances are correct, the evidence suggests that the original first letter was a particle, and it is removed, with the remainder of the string being returned. The string is returned unchanged if any of the new strings is incorrect. For instance, the word "لكتاب" (/b-kṯab/⟨to the book⟩) starts with the letter "ل", which belongs to the set of particles. Removing this letter leaves the word "كتاب" (/kṯab/⟨book⟩). Adding both "و" and "ف" to this word results in two valid instances, suggesting that the first letter is a particle.

Replicate and Remove (RPR). Our third algorithm performs two independent tests on a candidate string. First, the initial letter is removed, and the remaining word is checked against the lexicon. if it is not found, the original word is returned. Second, based on Rule 4.4 above, the initial letter is duplicated, and the result is tested for correctness against the lexicon. If test succeeds, the unchanged original word is returned (no stemming takes place).

We have noticed that if the word is a verb starting with "ب" *baa* or "ك" *kaa*, the first letter is removed whether it is a particle or not, since these are particles that cannot precede verbs. Duplicating them in verbs produces incorrect words, and causes the first letter of the original word to be removed. We can use the letter "ل" *lam* to recognise verbs that start with such particles. Accordingly, we introduce a new step where we add the letter "ل" *lam* to the word and check it for correctness. If the word is incorrect with the letter "ل" *lam*, and also incorrect with the first letter replicated, then we conclude that the word is not a verb, and we remove the first letter.

For words starting with the letter "ل" *lam*, we add both "ب" *baa* and "ك" *kaf* instead of replicating it, since replication might result in a correct word, and lead to the particle "ل" *lam* being preserved. If both new instances are incorrect, we remove the first "ل" *lam*.

Approach	Incorrect words	Possible particles	Altered words
RCL	82	17 037	146 032
RR	82	15 907	146 779
RPR	82	20 869	142 082

Table 4.6: Results of the new approaches, showing markedly fewer incorrect words, fewer possible particles, and comparable strength to the baseline in Table 4.5

For example, to remove the first particle from the word "كتاب" (/kt̪ab/⟨book⟩) using RPR, we first check the string "تاب" (/t̪ab/⟨rebent⟩) against the lexicon, which exists with totally different meaning. We duplicate the first letter and check the new word "ككتاب" (/kkt̪ab/⟨as the book of⟩) against the lexicon. As this word exists, no particle removal happens and we return the original word unchanged.

The above algorithms may be applied repeatedly. In particular, if stemming a word starting with either "و" *waw* or "ف" *faa* results in a new word of three or more characters that has either "و" *waw*, "ك" *kaf*, "ب" *baa*, or "ل" *lam* as its first character, the particle removal operation can be repeated; such repeated RPR is indicated as DRPR. Approaches may be combined; for example RC or RCL may be used with RPR, as many proper nouns and out-of-vocabulary-words start with particles that RPR and RR fail to remove.

Evaluation of Our Particle Removal Approaches

We have evaluated our new algorithms using the same data set described in Section 4.2.4. As seen from Table 4.6, all three algorithms result in a low number of incorrect words after stemming, with similar strength. However, they differ in the number of words with possible particles that remain after stemming. The RPR approach leaves many words with possible particles (around 5 000 more than the RR approach and 3 000 more than the RCL approach).

Our algorithms result in 82 incorrect words, compared to 5 164 using MT, 724 using RW, and 220 using RC. The number of words that start with possible particles drops dramatically with both RCL and RR. Although a fairer comparison of correctness could be carried out using a lexicon other than the one used to stem particles, our main objective of showing mistakes is to highlight the effects of removing particles without validating the remaining string, rather than the correctness of the stemmed words.

Using the RPR approach we extracted all unique words that have not been stemmed (words still having a first letter as a possible particle); these numbered 10 476 unique words.

	Stemmed using RPR		Stemmed using RR	
Word	Stem	Meaning	Stem	Meaning
وبطاقتي	بطاقتي	my ID card	طاقتي	my power
فاتتهم	فاتتهم	they missed it	اتتهم	it came to them
بوسادة	وسادة	pillow	سادة	masters
وليفي	وليفي	my mate	يفي	made his promise
وكفنها	كفنها	her coffin	فنها	her art
بوصفاتها	وصفاتها	her recipes	صفاتها	her characterstics

Table 4.7: Words with different meaning when stemmed by RPR and RR.

To check algorithm accuracy, we randomly selected 250 of these and examined them. We found that only 12 words included particles that we believe should be stemmed; this indicates an accuracy of around 95%.

As stemming particles can result in correct but completely different words, we decided to pass the list we extracted using the RPR approach to other approaches, and to check whether the stemmed words would be correct. We counted the number of correctly stemmed words changed by each approach. Of the 10 476 words, RR produced 4 864 new correct stems.

Manual examination of the output list of the RR algorithm shows some interesting trends. The algorithm achieves highly accurate particle recognition (few false positives). However, it often fails to recognise that the first letter is an actual particle, because replacing the first letter with "فـ" *faa* and "و" *waw* will often produce valid new words. For example, consider the word "بارع" (/baariʕ/⟨clever⟩). Applying the RR algorithm results in two valid words: "وارع" (/wʔrʕa/⟨and look after⟩), and "فارع" (/faʔrʕa/⟨and look after⟩). The first letter of the original word is therefore removed, giving the word "ارع" (/ʔrʕa/⟨look after⟩), instead of the original word "بارع".

We observe that about 90% of these are ambiguous, where the first character could be interpreted as a particle or a main character of the stemmed word; the meaning is different in the two cases. For example, the word "فيلم" (/fjlm/⟨film⟩) could also be read as "فَيَلُم" (/fajalum/⟨and he collects⟩), with the first letter read as a particle. MT, and RCL generated 3 950 similar stems, while RC returned 2 706 stems. Examples are shown in Table 4.7.

RPR keeps any letter that is possibly a core part of the word, even though it might also

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	0.576	**0.674**	**0.291**	**0.388**	**0.758**	**0.324**	**0.451**	**0.724**
wMT	0.391	0.576	0.656	0.284	0.364	0.656	0.320	0.435↓	0.656
wRR	0.390	0.560	0.657	0.288	0.382	0.657	0.322	0.441	0.657
wRCL	0.372	0.564	0.622	0.280	0.372	0.622	0.311↓	0.436	0.622
wRC	0.375	0.568	0.620	0.278	0.370	0.620	0.310↓	0.436	0.620
wRPR	**0.397**	0.564	0.685	0.287	0.380	0.685	0.323	0.441	0.685
wDRPR	0.396	0.564	0.685	0.287	0.380	0.685	0.323	0.441	0.685
wRPRRC	0.393	**0.588**	0.672	0.288	0.372	0.672	0.323	0.444	0.672
light10	0.416	0.644	0.641	0.350	0.438	0.838	0.372	0.507	0.757
wMT	0.425	0.640	0.628	0.350	**0.440**	0.836	0.375	0.507	0.751
wRR	0.414	0.628	0.623	0.351	0.434	0.840	0.372	0.499	0.751
wRCL	0.386	0.620	0.568	0.336	0.408	0.826	0.352	0.479	0.720
wRC	0.414	0.652	0.620	0.336	0.418	0.826	0.362	0.496	0.741
wRPR	0.425	0.648	0.655	0.351	0.434	0.840	0.376	0.505	0.764
wDRPR	0.425	0.640	0.655	0.350	0.432	0.840	0.375	0.501	0.764
wRPRRC	**0.433**	**0.672**	**0.660**	**0.353**	0.428	**0.843**	**0.380**↑	**0.509**	**0.768**

Table 4.8: Performance of particle removal algorithms: the top part shows results of integrating algorithms with the light10 stemmer without using relevance feedback, while the bottom part shows results of running the same algorithms along with relevance feedback. ↓ shows results significantly lower than the light10 stemmer, while ↑ shows results that are significantly better than those of light10 stemmer at the 95% confidence level.

be considered as a particle. In contrast, RR removes such letters. In most cases, keeping the letter appears to be the best choice.

Information Retrieval Evaluation While the ability to stem particles into valid Arabic words is valuable for tasks such as machine translation, document summarisation, and information extraction, stemming is usually applied with the intention of increasing the effectiveness of an information retrieval system. We therefore evaluate our approaches in the context of an ad-hoc retrieval experiment.

Table 4.8 shows the results recorded for each approach. Rows show the results of actuating individual particle removal techniques in the light10 stemmer. DRPR is repeated RPR, and RPRRC is RPR combined with RC. Results show that removing all particles decreases

the performance of the light10 stemmer. The MT technique results in a significant decrease in P@10 [t-test, $p = 0.032$], while RCL and RC result in a significant decrease in MAP [t-test, $p = 0.038$, and $p = 0.033$ respectively]. RPR, DRPR, and RPRRC slightly decrease P@10 and recall but not significantly. In contrast, when using relevance feedback, MT, RPR, DRPR, and RPRRC improve the light10 results; however, only RPRRC adds a significant improvement in MAP [t-test, $p = 0.040$]. RPR and DRPR add a weakly significant improvement in terms of recall [t-test, $p = 0.067$, and $p = 0.072$ respectively]. RC, RR, and RCL perform below the baseline, but not significantly so.

In conclusion, uncontrolled removal of Arabic particles should be avoided, as this decreases precision. The best alternative is to use a combination of RPR for removing particles from Arabic words, and to use RC for removing particles from words that do not exist in the lexicon (for example, proper nouns and foreign words). RCL and RR work the same; however, RCL is recommended as RR introduces more stemming errors than RCL.

Arabic has more prefixes than particles. We continue in the next two subsections with a discussion of how to stem the definite article and verb prefixes.

4.2.5 Stemming the Prefix "ال"

The most well-known prefix for nouns is the definite article "ال" (/ʔl/⟨the⟩). All existing retrieval stemmers stem this prefix in a similar manner [Khoja and Garside, 1999; Darwish and Oard, 2002; Larkey et al., 2002], by comparing the first letters of the word with this prefix without any validation. Since particles can be removed using one of the above approaches, removing "ال" would be generally straightforward; however, we need to be careful about proper nouns and words that start with "ال" as core letters.

Words with the patterns "اِفعَال" and "اِفتِعَال", such as "اِلهَام" (/ʔilhaam/⟨inspiration⟩) and "اِلتِهَاب" (/ʔilṭihaab/⟨infection⟩) should not be stemmed, since the letters "ال" are a core part of such words. Stemming these letters would cause the words to change in meaning, placing them under different index references, and in some cases leaving them meaningless. For example, removing this prefix from these two words leaves "هَام" (/haam/⟨important⟩) and "تِهَاب" (/ṭihaab/⟨meaningless⟩).

In Arabic a noun that accepts the definite article should not accept another definite article as another prefix. Formally, based on the assumptions presented in Section 4.2.4, let al be the Arabic definite article, and w be any word in L. Then:

- Rule 5: A word that starts with the definite articles as a prefix should not accept another definite article.

$$\forall(al + w) \in L \Rightarrow (al + (al + w)) \notin L \tag{4.5}$$

Based on this rule, we remove the first "الـ" *al* only if adding another "الـ" to a valid Arabic word that starts with *al* forms an invalid Arabic word. We call this rule "Strict AL "الـ" Removal (SAL)". This approach therefore avoids the incorrect removal of the core "الـ" letters.

Another category of words that might have an "الـ" at their beginning are proper nouns. Most AIR stemmers stem the combination "كالـ", "فالـ", "والـ", "للـ", and "بالـ". There are many proper nouns, especially foreign words, that start with this combination. We remove such a prefix by verifying that the remaining string exists in the lexicon before removing such a combination. We name this rule "Remove al Combinations (RalC)". Table 4.9 shows the result of both techniques compared to the baseline; noAL refers to light10 without stemming the definite article "الـ".

It is clear that stemming "الـ" improves retrieval effectiveness; noAl is significantly worse than light10 used with any technique to stem the definite article. Adding RalC to light10 does not significantly affect effectiveness, with or without relevance feedback. Combining SAL with light10 does not help effectiveness, and leads to poorer results with relevance feedback.

4.2.6 Stemming Verb Prefixes

We introduce rules to stem verb prefixes based on the Arabic grammar as follows:

Let L be an Arabic lexicon, P be the set of the two prepositions {"كـ", "بـ"}, VP be the set of verb prefixes {"ا", "نـ", "يـ", "تـ", "سـ"}, N be the set of nouns in the lexicon, V be the set of verbs, p be a preposition in P, al be the Arabic definite article, and w be any word in L. Then:

- Rule 6: Only a noun accepts the definite article and prepositions.

$$\forall(w \in L) \wedge (al + w \in L) \Rightarrow w \in N \tag{4.6}$$

and

$$\forall(w \in L) \wedge (p + w \in L) \Rightarrow w \in N$$

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	**0.576**	0.674	**0.291**	**0.388**	**0.758**	0.324	**0.451**	0.724
noAL	0.327	0.556	0.581	0.244	0.304	0.710	0.272↓	0.388↓	0.657↓
wSAL	0.393	0.568	0.680	0.289	0.376	**0.758**	0.324	0.440	**0.726**
wRalC	**0.395**	0.572	**0.681**	**0.291**	0.384	0.757	**0.325**	0.447	**0.726**
light10	**0.416**	0.644	0.641	0.350	**0.438**	0.838	0.372	0.507	0.757
noAL	0.405	**0.680**	**0.647**	0.295	0.352	0.799	0.332↓	0.461↓	0.737↓
wSAL	0.404	0.648	0.612	**0.352**	0.432	0.839	0.369	0.504	0.746
wRalC	**0.416**	0.648	0.640	0.351	0.436	**0.840**	**0.373**	**0.507**	**0.758**

Table 4.9: Effects of removing "ال". Removing "ال" is better than not removing it at all. RaLC improves stemming better than the SAL technique. ↓ represents significantly different results at the 95% confidence level than the light10 performance.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	**0.391**	0.576	0.674	0.291	**0.388**	**0.758**	**0.324**	**0.451**	**0.724**
noVerbs	**0.391**	**0.580**	**0.677**	0.241	0.332	0.654	0.291	0.415	0.664
wVerbsToPast	0.386	0.560	0.670	**0.292**	0.382	**0.758**	0.323	0.441	0.722
light10	0.416	0.644	0.641	0.350	**0.438**	**0.838**	0.372	**0.507**	**0.757**
noVerbs	**0.425**	0.628	**0.670**	0.302	0.370	0.729	0.343	0.456	0.705
wVerbsToPast	0.421	**0.648**	0.645	**0.351**	0.428	0.835	**0.374**	0.501	**0.757**

Table 4.10: Effects of stemming verb prefixes on retrieval; stemming verb prefixes is better than not stemming them.

- Rule 7: Based on Rule 6, a word is not a noun (and might be a verb) if it does not accept either a preposition or the definite article "ال":

$$\forall(w \in L) \wedge (p + w) \notin L \Rightarrow (w \notin N) \tag{4.7}$$

and

$$\forall(w \in L) \wedge (al + w) \notin L \Rightarrow (w \notin N)$$

- Rule 8: We conclude that a word is a verb if it is not a noun as in Rule 7 and starts with verb prefixes:

$$\forall w \notin N \wedge w_1 \in VP \Rightarrow w \in V \tag{4.8}$$

Based on these rules, we stem words like "سِيارَة" (/sijaraa/⟨a car⟩), "سِتارَة" (/siṯaraa/⟨a curtain⟩), and "سارَة" (/saraa/⟨the proper noun Sarah⟩) by checking whether they are verbs before removing the prefixes "سيـ", "ستـ" and "سا" respectively. Removing verb prefixes causes most verbs revert to the past tense. There is one category, known as the hollow verbs, that instead reverts to the infinitive form. These contain verbs with long vowels in the middle of the three-letter past tense verb. When stemming verbs, these should be treated separately and returned to their past form by replacing the middle vowel with the vowel alef "ا". Examples of this class are: "يقول" (/jqwl/⟨says⟩), and "يموت" (/jmwṯ/⟨kills⟩). Stemming the prefix "يـ" in those verbs would leave "قَول" (/qawl/⟨a say⟩), and "مَوت" (/mawṯ/⟨death⟩) respectively. Replacing the middle vowel with "ا" would leave "قَالَ" (/qaala/⟨said⟩) and "ماتَ" (/maṯa/⟨died⟩), which are the past tense forms.

We checked the effects of both removing verbs completely from the collection and the queries, and stemming verbs on retrieval. The light10 stemmer does not stem any verb prefixes. Results are shown in Table 4.10. In this table, noVerbs stands for not indexing verbs at all, and wVerbsToPast stands for stemming verbs and returning and modifying hollow verbs. We see that stemming verbs adds some improvements to precision over the light10 stemmer. Not indexing verbs at all negatively affects the retrieval performance.

4.2.7 Overall Prefix Removal Approach

The above prefix removal techniques are added altogether to the light10 stemmer in the following order,

- Remove the prepositions using the RPRRC technique.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	**0.391**	0.576	**0.674**	0.291	**0.388**	0.758	**0.324**	**0.451**	**0.724**
wPrfxR	0.389	0.568	0.671	0.291	0.368	0.761	0.323	0.435	**0.724**
wNorPrfxR	0.371	**0.588**	0.627	**0.293**	0.372	**0.766**	0.319	0.444	0.709
light10	0.416	0.644	0.641	0.350	**0.438**	0.838	0.372	0.507	0.757
wPrfxR	0.402	0.636	0.588	0.351	0.426	0.837	0.368	0.496	0.735
wNorPrfxR	**0.442**	**0.688**	**0.697**	**0.355**	0.422	**0.840**	**0.384↑**	**0.511**	**0.782**

Table 4.11: Effects of adding our normalisation and prefix removal techniques to the light10 stemmer. Prefix removal alone has a negative effect on retrieval effectiveness, but the reverse is true when it is used in conjunction with normalisation and relevance feedback.

- Remove the definite article using the SaL and RalC techniques.
- Remove suffixes as in the light10 stemmer.
- Remove verb prefixes.

We refer to this technique as "wPrfxR". Table 4.11 shows the results of running the light10 stemmer with our prefix removal techniques. This prefix removal technique has a negative impact. Combining this prefix removal and normalisation (wNorPrfxR) also reduces the performance of the light10 stemmer. In contrast, using relevance feedback results in a significant improvement in MAP [t-test, $p = 0.043$], and also improves P@10 and recall.

Arabic has more suffixes than prefixes. Some suffixes are easily recognised and rarely occur as core letters of Arabic words. Longer suffixes are easier to stem than shorter ones [Aljlayl and Frieder, 2002]. We stem suffixes differently and use different rules based on the suffix type. In the following subsections, we present how to stem these suffixes and their individual effect on stemming using light10 as a baseline, and then present the effect on stemming of using a combined approach.

4.2.8 Possessive Pronouns Suffixes

We start by removing personal and possessive pronouns suffixes that are longer than one character; these are "هم" (/hm/⟨their⟩), "ها" (/haa/⟨her⟩), and "هما" (/huma/⟨their -dual-⟩). We do not stem the second-person pronouns as their frequency in written text is very low. In

the TREC 2001 collection, 158 647 words found with the pronouns "كم" (/kum/⟨yours -plural-⟩) and "كما" (/kuma/⟨yours -dual-⟩); indeed, "كم" (/kam/⟨as or and⟩); appears far more frequently, 71 076 words, followed by 60 373 words with the sequence "حُكم" (/ħukum/⟨ruling⟩) as core letters. Removing such suffixes might affect other words that originally end with these suffixes. Third-person suffixes occur more frequently in written text and should all be removed. To check whether a suffix is a pronoun, we replace it with other pronouns and check whether the resultant word is correct. A valid word that ends with the suffix "ها" should be also valid when this suffix is replaced with "ه", "هم", or "هن". Let PS be the set of personal and possessive pronouns, and ps_1 and ps_2 be a suffix in PS. We remove these suffixes based on the following rule.

- Rule 9: A word that accepts a personal or a pronoun suffix, should accept other personal and pronoun suffixes.

$$\forall (w \in L) \wedge (w + ps_1) \in L \Rightarrow (w + ps_2) \in L \tag{4.9}$$

The light10 stemmer removes only the pronoun suffix "ها" and "ـه". Table 4.12 shows the effects of stemming these suffixes on the light10 stemmer. The label "wPP" refers to stemming the third-person pronouns mentioned above, and "noPP" refers to not stemming pronouns at all. Results show that stemming these suffixes does not produce a major effect on retrieval results with or without relevance feedback. It does increase recall, but not significantly. Similar performance is obtained using noPP with a slight improvement in precision and recall.

4.2.9 The Dual Suffix "ان"

The suffix "ان" occurs frequently in Arabic words, but these letters may also be a core part of a word, for example in the word "بُستَان" (/bus̱taan/⟨garden⟩). To stem this suffix while avoiding stemming core letters, we use the grammatical rule that a dual terminates with "ان" when it is in the nominative mode, and in "ين" or "ا" when it is in the accusative and genitive moods. We replace the last "ان" with "ين" or "ا" and check the new word in the Arabic lexicon; we stem this suffix only if the new instance of the word exists.

If the above rules fail to remove this suffix, the prefix should be checked to see whether "ان" is a verb suffix. We remove it if the prefix is either "يـ" or "سيـ" (verb prefixes). The suffix could also be removed by checking for the remaining string in the lexicon. If the

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	**0.576**	0.674	0.291	**0.388**	**0.758**	0.324	0.451	0.724
noPP	**0.396**	0.572	**0.684**	**0.292**	0.382	0.757	**0.326**	**0.445**	**0.727**
wPP	0.395	0.564	0.682	**0.292**	0.386	**0.758**	**0.326**	**0.445**	**0.727**
light10	0.416	0.644	0.641	**0.350**	0.438	**0.838**	**0.372**	**0.507**	0.757
noPP	**0.426**	**0.648**	**0.666**	0.345	0.432	**0.838**	**0.372**	0.504	**0.768**
wPP	**0.426**	0.640	**0.666**	0.345	**0.440**	0.834	**0.372**	**0.507**	0.765

Table 4.12: Effects of stemming pronoun suffixes. Stemming pronoun suffixes has no affect on light10 retrieval performance.

remaining string is not found we retain it. The latter rule sometimes fails as there are cases where removal of this suffix from some non-dual words creates a valid word. For example, stemming the proper noun "نُعمان" /nuʕman/ results in "نعم" (/nʕm/⟨yes⟩). These two rules can be formalised based on the previous assumptions and having *DS* as the set of dual suffixes {"ان","ين", "ا"} and letting ds_1, ds_2 be any two elements in *DS* then:

- Rule 10: A valid dual word that ends with "ان" should also be valid when it ends with "ين" or "ا".

$$\forall(w + ds_1) \in L \Rightarrow (w + ds_2) \in L \tag{4.10}$$

The light10 stemmer removes this suffix without any checking. As seen from Table 4.13, not stemming this suffix (noAN), or stemming the suffix using our rules (wAN) are both more effective than the light10 approach.

4.2.10 The Suffix "ات"

This suffix indicates the feminine sound plural as described in Section 2.1.4. It is usually an extension to the final *taa marbuta* "ة", and can be stemmed if the remaining string exists in the lexicon. If the remaining string is not a valid Arabic word, further checking after replacing it with *taa Marbuta* "ة" assures that the suffix is the feminine sound plural. Formally, let *at* represent the suffix "ات"; *st* represent the suffix "ة", then we can derive Rule 11 as:

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	**0.576**	0.674	0.291	**0.388**	0.758	0.324	**0.451**	0.724
noAN	0.393	0.560	0.677	**0.295**	0.380	**0.759**	**0.328**	0.440	0.725
wAN	**0.395**	0.568	**0.682**	0.293	0.386	0.757	0.327	0.447	**0.727**
light10	0.416	0.644	0.641	**0.350**	**0.438**	**0.838**	**0.372**	**0.507**	0.757
noAN	0.422	**0.648**	0.650	0.347	0.426	0.834	**0.372**	0.500	0.759
wAN	**0.426**	0.640	**0.658**	0.345	**0.438**	0.832	**0.372**	0.505	**0.761**

Table 4.13: Effects of stemming the suffix "ان". Stemming this suffix using our rules, or not stemming it at all, is better than stemming it without validation.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	**0.576**	0.674	**0.291**	**0.388**	**0.758**	0.324	**0.451**	0.724
noAT	0.365	0.556	**0.682**	0.281	0.376	0.755	0.309	0.436	0.725
wAT	**0.395**	0.568	**0.682**	**0.291**	0.382	0.757	**0.325**	0.444	**0.727**
light10	0.416	**0.644**	0.641	**0.350**	0.438	**0.838**	**0.372**	**0.507**	0.757
noAT	0.384	0.632	0.655	0.327	0.410	0.831	0.346	0.484	0.759
wAT	**0.425**	0.640	**0.656**	0.346	**0.440**	0.835	**0.372**	**0.507**	**0.762**

Table 4.14: Effects of stemming "ات. Stemming the suffix "ات" with or without rules has similar effects on retrieval, but not stemming it negatively affects retrieval performance.

- Rule 11: A valid Arabic word that ends with the two letters "ات" as the feminine sound plural suffix should also be valid when replacing "ات" with the feminine singular suffix "ة".

$$\forall(w + at) \in L \wedge (w + st) \in L \Rightarrow w \in L \tag{4.11}$$

Based on this rule, the suffix "ات" can be removed if the word exists in the normal singular feminine form with the suffix "ـة".

Table 4.14 shows that stemming this suffix using our rules does not have an effect on the stemming result, indicating that stemming this suffix without any rules does not have negative effects on retrieval. At the same time, removing this suffix is shown to be better than leaving it in place.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	**0.576**	0.674	0.291	**0.388**	0.758	0.324	**0.451**	0.724
noWNYN	**0.399**	0.572	0.680	0.289	0.378	0.750	0.326	0.443	0.721
wWNYN	0.397	**0.576**	**0.684**	**0.295**	0.386	**0.763**	**0.329**	0.449	**0.731**
light10	0.416	0.644	0.641	**0.350**	**0.438**	**0.838**	0.372	**0.507**	0.757
noWNYN	**0.431**	0.644	**0.676**	0.342	0.422	0.835	0.371	0.496	**0.770**
wWNYN	0.425	**0.648**	0.657	0.349	0.436	0.836	**0.374**	**0.507**	0.762

Table 4.15: Effects of stemming "ون" and "ين"; the light10 stemmer stems these suffixes without any verification rules. Using the light10 stemmer without stemming these suffixes (noWNYN), or using the light10 stemmer and applying our rules to stem these suffixes, both produce better results than the default light10 stemmer.

4.2.11 The Suffixes "ون" and "ين"

These suffixes are added to the singular form of the noun to indicate the masculine sound plural. The same rule used to stem the "ان" suffix can be applied to these suffixes. The suffix "ون" can be recognised by replacing it with "ين", and stemming only if this yields a valid word. The suffix "ون" is also a suffix of the imperfect verbs "الافعال الخمسة". In cases where replacing the suffix "ون" with "ين" results in an incorrect word in the lexicon, the word prefixes can be checked; if the word starts with either "ي" or "سيـ" (verb prefixes), then this suffix can be safely removed. Likewise, the suffix "ين" can be removed if replacing it with either "ون" or "ان" results in a correct word in the lexicon.

Table 4.15 shows the results obtained by stemming these two suffixes using our rules. Using our rules, the result shows a slight improvement over the light10 stemmer, which stems them without any validation. The increase is similar with and without relevance feedback.

4.2.12 The Single Letter Suffixes "ـة" and "ي"

The suffix "ة" is very common. In the titles fields of TREC 2001 and TREC 2002 queries, 52 (20%) words out of 208 unique words end with this suffix. Before stemming this suffix, we must check whether the remaining string is correct, while avoiding conflating words that differ in meaning. For example, the word "الجامِعَة" (/alʒamiʕaa/⟨the university⟩) would be stemmed to "الجامِع" (/alʒamiʕ/⟨the mosque⟩), which is naturally a valid Arabic word. We do

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	**0.391**	**0.576**	**0.674**	**0.291**	**0.388**	**0.758**	**0.324**	**0.451**	**0.724**
noSingle	0.321	0.556	0.623	0.259	0.354	0.720	0.280↓	0.421	0.680
wSingle	0.372	0.572	0.647	0.288	0.382	0.745	0.316	0.445	0.705
light10	**0.416**	0.644	0.641	**0.350**	**0.438**	**0.838**	**0.372**	**0.507**	0.757
noSingle	0.386	**0.652**	**0.698**	0.317	0.384	0.821	0.340	0.473	**0.771**
wSingle	0.405	0.648	0.650	0.345	0.422	0.833	0.365	0.497	0.758

Table 4.16: Effects of stemming single character suffixes. noSingle shows the results of light10 stemmer without stemming suffixes "ة" and "ي", wSingle shows results of stemming using light10 with our new rules. Stemming this suffixes using rules decreases the performance of the light10 stemmer, but leaving it without stemming results in a significant decrease in results.

not handle such cases as these need techniques such as word disambiguation and considering other words in the sentence. We minimise stemming mistakes by returning the letter "ت" to its origin letter "ة" after stemming. We either stem the new "ة" suffix, or leave it if the remaining string is not a valid word.

The suffix "ي" could be the relative suffix or the subject (first person) possessive pronoun, as in the words "ليبي" (/lijbj/⟨Libyan⟩) and "كِتابي" (/kiṯabj/⟨my book⟩) respectively. Stemming the suffix in the first case will not bring the word to its class terms in many cases, for example, removing the suffix in the word "ليبي" will leave the string "ليب" which is not among the terms that reference words that relate to Libya. As such, the suffix should be removed only if the remaining word is valid. In the second case, the suffix can be properly removed be replacing it with one of the third-person possessive pronouns "ها" ⟨her⟩ and "ه" ⟨his⟩, and ensuring that the two new instances are correct words in the lexicon.

Table 4.16 shows the results of stemming these suffixes with the light10 stemmer. Both precision and recall are negatively affected by removing these suffixes (indicated by "wSingle"), indicating the difficulty of stemming these suffixes. Leaving these suffixes without stemming (indicated by "noSingle") worsens the results, suggesting that stemming is better.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	**0.391**	0.576	**0.674**	0.291	**0.388**	0.758	0.324	**0.451**	**0.724**
noSfxR	0.289	0.564	0.568	0.238	0.330	0.683	0.255↓	0.408	0.636↓
wSfxR	0.360	**0.600**	0.628	0.275	0.364	0.739	0.303	0.443	0.693
NorPrfxRSfxR	0.384	0.588	0.647	**0.297**	0.368	**0.768**	**0.326**	0.441	0.718
light10	0.416	0.644	0.641	0.350	0.438	0.838	0.372	0.507	0.757
noSfxR	0.354	0.636	0.652	0.290	0.378	0.794	0.312↓	0.464	0.736↓
wSfxR	0.417	0.656	**0.688**	0.333	0.422	0.829	0.361	0.500	0.771
NorPrfxRSfxR	**0.429**	**0.676**	0.687	**0.358**	**0.444**	**0.839**	**0.382**	**0.521**	**0.776**

Table 4.17: Effects of stemming All suffixes; leaving suffixes without removal "noSfxR" significantly reduces retrieval performance; while adding our new suffix removal techniques reduces the effectiveness of the light10 stemmer. Adding our normalisation, prefix removal techniques, and suffix removal techniques improves the performance of the light10 stemmer. ↓ represents significantly different results at the 95% confidence level than the light10 performance.

4.2.13 Overall Suffix Removal

Table 4.17 shows the effects of adding all the above suffix removal techniques to the light10 stemmer. Results show that adding suffix removal techniques only "light10SfxR" reduces retrieval performance. This is as expected, since removing all suffixes without any reference to prefixes is inadvisable. The light10 stemmer removes only the particle "و" and the definite articles, and leaves other prefixes such as verb prefixes and other inseparable particles. When adding other prefix removal techniques to the light10 stemmer "NorPrfxRSfxR", results improve. However, the differences are not statistically significant.

4.2.14 Our New Stemmers

We present our stemmers based on our experiments in the previous subsections. We introduce two types of algorithms: a rule-based stemmer where we use our rules to validate stemming affixes, and light stemmers where we modify the number of affixes removed in the light10 stemmer.

Rule-based Light Stemmers

We integrate all our techniques into the light10 stemmer to explore their impact on retrieval performance.

We call our algorithms, “Restrict”, “Restrict1”, and ”Restrict2”.

In the “Restrict” algorithm we:

- normalise the text as in Section 4.2.2 except that we do not join compound nouns.
- remove stopwords as in the light10 stemmer,
- remove prepositions and conjunctions using the RPR technique,
- remove the definite article if the remaining string contains three or more characters,
- remove the verb prefixes,
- remove pronoun suffixes “ها”, “هم”, “هما”, and “هن”,
- remove the suffixes “ات”, “ون”, “ان”, “ين” using our rules,
- remove the suffix “ه”, “يه”, and “وا”, and
- remove the single suffix “ي” if replacing it with “ه” or “ها” results in a correct word.

In the “Restrict1” algorithm we:

- normalise the collection fully as described in Section 4.2.2,
- remove prepositions and conjunctions using our RPRRC technique,
- remove the definite articles using our SAL and RalC techniques,
- remove verb prefixes, and
- remove suffixes as in the light10 stemmer.

In the “Restrict2” algorithm we replace the original light10 suffix removal techniques in “Restrict1” with our new suffix removal techniques.

Results of these three algorithms are shown in Table 4.18. Restrict and Restrict2 slightly increase the performance of the light10 stemmer, while Restrict1 decreases the performance of this stemmer.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	0.576	0.674	0.291	**0.388**	0.758	0.324	**0.451**	0.724
Restrict	**0.401**	0.576	**0.685**	0.290	0.376	0.762	**0.327**	0.443	**0.730**
Restrict1	0.371	**0.588**	0.627	0.293	0.372	0.766	0.319	0.444	0.709
Restrict2	0.384	**0.588**	0.647	**0.297**	0.368	**0.768**	0.326	0.441	0.718
light10	0.416	0.644	0.641	0.350	0.438	0.838	0.372	0.507	0.757
Restrict	**0.445**	0.676	**0.713**	0.352	0.442	0.835	0.383	0.520	**0.784**↑
Restrict1	0.442	**0.688**	0.697	0.355	0.422	**0.840**	**0.384**↑	0.511	0.782↑
Restrict2	0.429	0.676	0.687	**0.358**	**0.444**	0.839	0.382	**0.521**	0.776

Table 4.18: Performance of our three algorithms. The upper half shows performance without the relevance feedback, while the lower half shows the performance with relevance feedback. While our algorithms show similar performance to the baseline without using relevance feedback, they show significant improvement with relevance feedback.

When using relevance feedback (at the lower half of Table 4.18), our algorithms outperform the light10 stemmer. The increase in recall by Restrict and Restrict1 is statistically significant [t-test, $p = 0.039$, and $p = 0.012$ respectively]. In terms of MAP, Restrict is significantly better [t-test, $p = 0.043$], while the improvement for Restrict1 is weakly significant [t-test, $p = 0.071$]. Restrict2 performs better than the light10 stemmer, but the improvement is not significant for MAP, P@10, or recall.

More Light Stemmers

Based on the results in the previous section, we also draw up three lighter stemmers out of the light10 stemmer that we call "light11", "light12", and "light13".

In the "light11" stemmer, we omit pronoun suffix stemming, as well as the unnecessary steps in the light10 stemmer where the suffixes "ة" and "ية" are stemmed after normalisation. We also reduced the stopword list by removing non-normalised words, as the light10 stemmer normalises words before matching words with the stopword list. This results in 122 stopwords instead of 168 words.

In the "light12" stemmer, we omit more suffixes and remove only "ات","ه", "ي", and "يه". We also normalise words before stemming using our normalisation techniques without

	Prefixes Removed	Suffixes Removed
light10	للـ، الـ، والـ، بالـ، كالـ، فالـ، و	ها، ان، ات، ون، ين، يه، ية، ه، ة، ي
light11	للـ، الـ، والـ، بالـ، كالـ، فالـ، و	ان، ات، ون، ين، يه، ه، ي
light12	للـ، الـ، والـ، بالـ، كالـ، فالـ، و	ات، يه، ه، ي
light13	و، بـ، كـ، فـ، لـ، للـ، الـ	ات، يه، ه، ي

Table 4.19: Affixes removed by light10, light11, light12, and light13 stemmers.

	TREC 2001			**TREC 2002**			**TREC 2001 and 2002**		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	0.391	0.576	0.674	0.291	**0.388**	**0.758**	0.324	0.451	0.724
light11	0.396	0.572	**0.684**	0.292	0.382	0.757	0.326	0.445	**0.727**
light12	0.399	0.584	0.680	**0.294**	0.380	0.753	**0.329**	**0.448**	0.723
light13	**0.401**	**0.596**	0.680	0.290	0.374	0.753	0.327	**0.448**	0.723
light10	0.416	0.644	0.641	0.350	0.438	0.838	0.372	0.507	0.757
light11	0.426	**0.648**	0.666	0.345	0.432	0.838	0.372	0.504	0.768
light12	**0.431**	**0.648**	**0.678**	0.350	0.436	0.833	0.377	0.507	0.769
light13	0.430	0.644	0.668	**0.352**	**0.440**	**0.846**	**0.378**	**0.508**	**0.773**↑

Table 4.20: Results of the light11, light12 and light13 stemmers compared to light10 stemmers. The new light stemmer perform better than the light10 stemmer with and without relevance feedback. ↑ shows values that are significantly better than the light10 stemmer at the 95% confidence level.

splitting words.

In the "light13" stemmer, we remove all prepositions and conjunctions using the RPRRC technique, and then we remove the definite article "الـ" as well as removing the same suffixes removed by the light12 stemmer.

Table 4.19 shows the prefixes and suffixes removed by the original light10, light11, light12, and light13 stemmers.

As can be seen from Table 4.20, the new stemmers perform better than the light10 baseline; light13 adds a significant improvement in recall [t-test, $p = 0.029$] when using relevance feedback; and light11 adds a weakly significant improvement in recall [t-test, $p = 0.076$]. None of the algorithms lead to a significant improvement in MAP or P@10.

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
Restrict2	**0.384**	**0.588**	**0.647**	**0.297**	0.368	**0.768**	**0.326**	**0.441**	**0.718**
Restrict2E	0.380	0.576	0.646	0.291	0.368	0.758	0.320↓	0.437	0.712↓
Restrict2	**0.429**	**0.676**	**0.687**	**0.358**	**0.444**	**0.839**	**0.382**	**0.521**	**0.776**
Restrict2E	0.424	0.668	0.681	0.346	0.426	0.824	0.372↓	0.507	0.765

Table 4.21: Performance of Restrict2 using extracted Office 2003 lexicon words.

4.2.15 Using the Collection as a Lexicon

Our approach depends on validation before removing affixes. This requires a comprehensive Arabic lexicon that contains the different forms of Arabic words. In the above stemmers we used the Microsoft Office 2003 lexicon. This affects the efficiency and the portability of our stemmers, since stemming the whole collection takes over 100 minutes and cannot be carried out on machines that do not run Microsoft Windows with the Office 2003 application suit installed. In this section we describe two alternative options.

Using the Extracted Office Lexicon

One option is to extract a list of words from the Microsoft Office lexicon. To do this, we passed all combinations of one to seven Arabic letters to the lexicon, and determined which words were valid; 2 976 645 words remained. We did not process words larger than 7 characters to maintain tractability. We minimised the number of words to test by excluding character sequences that are not valid in Arabic. For example, we eliminated sequences of three identical characters.

Table 4.21 shows the results obtained using this extracted lexicon. Restrict2 with the extracted lexicon — which we name Restrict2E — performs significantly worse than the original Restrict2, which uses the full Microsoft Office 2003 lexicon. This result is expected, as many Arabic words inflect to more than seven characters. Such words are not stemmed as they are considered OOV, and most of our rules will not apply to them.

Using the Corpus as a Lexicon

Corpora usually have the different forms of words used regularly, and we believe that using corpora terms as a lexicon would be sufficient for our stemmer. To generate the corpus

	TREC 2001			TREC 2002			TREC 2001 and 2002		
	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
light10	**0.391**	0.576	**0.674**	0.291	0.388	0.758	0.324	0.451	**0.724**
Restrictc	0.388	0.564	0.662	0.296	0.388	0.768	0.327	0.447	**0.724**
Restrict1c	0.384	0.552	0.663	0.292	0.378	0.762	0.322	0.436	0.721
Restrict2c	0.384	**0.620**	0.642	**0.300**	**0.390**	**0.772**	**0.328**	**0.467**	0.719
light10	0.416	0.644	0.641	0.350	0.438	0.838	0.372	0.507	0.757
Restrictc	0.407	0.628	0.583	0.296	0.388	0.769	0.373	0.497	0.737
Restrict1c	0.437	0.668	0.684	0.361	0.438	0.843	0.386↑	0.515	0.778
Restrict2c	**0.444**	**0.680**	**0.721**	**0.362**	**0.450**	**0.844**	**0.390↑**	**0.527**	**0.793↑**

Table 4.22: Effects of using the unique terms of the corpus as a lexicon. Stemming results are better than when using the Microsoft Office 2003 lexicon.

lexicon, we extracted all unique words of the TREC 2001 corpus, and repeated the Restrict, Restrict1 and Restrict2 experiments using this new lexicon rather than the Microsoft Office 2003.

Table 4.22 shows results obtained using this lexicon. Our algorithms are as good as the light10 stemmer, with Restrictc and Restrict2c performing slightly better. In contrast, when using relevance feedback, Restrict1c improves MAP significantly [t-test, $p = 0.017$]; and Restrict2c improves recall and MAP [t-test, $p = 0.036$, and $p = 0.040$ respectively]. The success of using the corpus with our rules is due to the fact that our techniques remove affixes better when using the actual terms of the corpus. An example that does not work when using professionally prepared Arabic lexicons is stripping prefixes from words that are not in that lexicon, such as foreign words. By using the corpus as lexicon, our approaches — such as the preposition removal techniques described in 4.2.4 — are more effective even with proper nouns and foreign words. For example, the technique RPR that strips conjunctions and prepositions from Arabic words works only when the remaining string after the first character is found in the lexicon. Such proper nouns and foreign words frequently do exist in the corpus, but may not exist in the professional lexicon. This enables the technique to correctly strip the first letter if it is a conjunction or a preposition. For instance, the word "وكاليفورنيا" (/wkaljfwrnja/⟨and California⟩) is a foreign word with the conjunction "و". Using our RPR technique with the Microsoft Office 2003 lexicon leaves the word untouched, as the proper noun "كاليفورنيا" does not exist in that lexicon. However, when using the

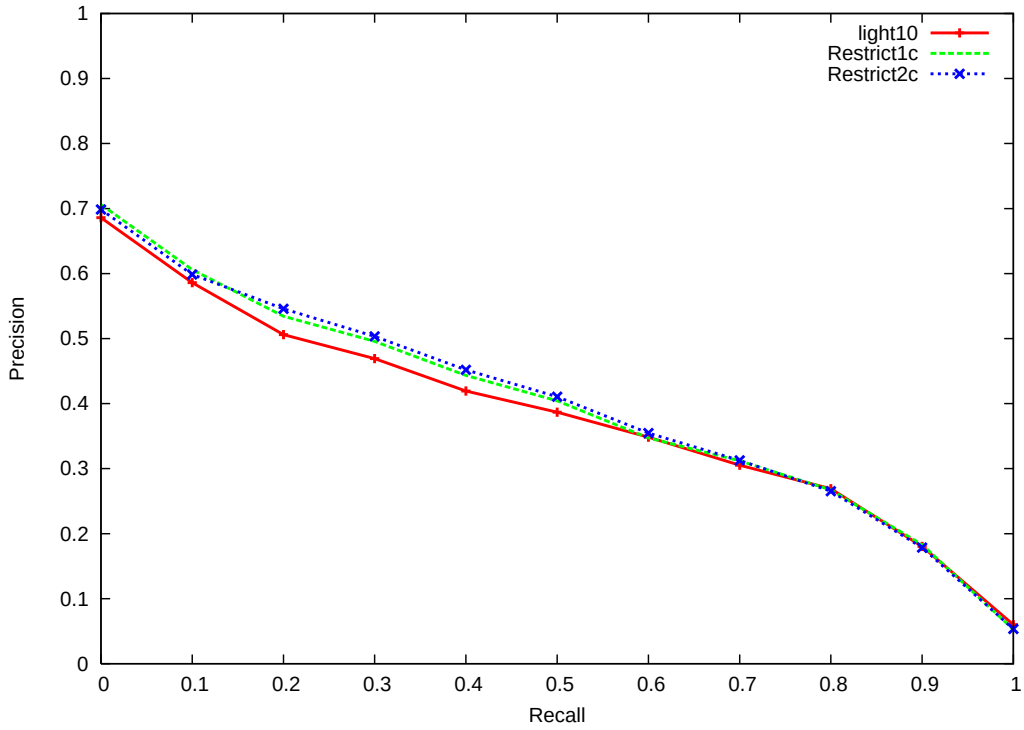

Figure 4.3: Performance of the light10 plus the normalisation and the prefix removal techniques on the 11 recall points.

corpus terms lexicon, the stemmer correctly returns "كاليفورنيا" without the conjunction "و", as the proper noun alone does exist in the corpus. Using the corpus improves not only retrieval performance for our rule-base light stemmers, but also improves their efficiency. The Restrict1c algorithm stems the collection in 11 minutes. It is about 5.5 times faster than using the Microsoft Office 2003 lexicon (61 minutes), 3 times faster than the Buckwalter stemmer (36 minutes), and only 1.5 times slower than the light10 stemmer (7.5 minutes). Figure 4.3 shows the performance of these two algorithms along with the baseline light10 stemmer. Our algorithms conflate Arabic terms better than algorithms that do not use morphological rules. Table 4.23 shows the number of terms, unique terms and index size for all stemming algorithms described.

4.2.16 Concluding Remarks

The experiments carried out in the last section show that light stemming in general can

	Index		
	Terms	Unique terms	Size (Kilobytes)
Khoja	59 279 415	193 932	482 376
B.Lemma	57 247 663	206 370	477 348
Restrict1	57 475 134	230 919	487 032
Restrict1c	57 478 755	233 062	488 096
Restrict2c	57 465 279	247 515	488 672
Al-StemN	57 525 479	252 526	492 580
Restrictc	59 098 485	254 705	502 468
B.Stem	57 521 776	258 059	487 336
Restrict2	57 450 603	273 620	491 364
light13	57 566 788	277 601	497 296
light10	57 621 607	279 194	496 812
Restrict	55 821 287	285 306	480 568
B.Stem2	57 662 918	292 763	495 908
light11	55 820 872	295 544	484 652
light12	57 568 036	321 810	502 960
noStemming	71 769 922	523 727	644 120
Al-Stem	71 977 278	624 809	640 952

Table 4.23: Number of terms, unique terms, and index size of each algorithm. Algorithms that use rules produce fewer unique terms than those that do not use rules in stemming. Stemmers are ordered by their unique terms.

be improved further. Our results show that retrieval using the TREC 2001 test collection is more susceptible to improvements when using morphological rules than the TREC 2002 collection. This could be to related to:

- The number of queries used in each collection: TREC 2002 has more queries than TREC 2001 (50 and 25, respectively).
- The difference in the way that relevance judgements were created, as discussed in Section 2.3.2.
- The nature of queries in both collections. TREC 2001 queries contain few proper nouns or foreign names, while TREC 2002 queries contain these more frequently. Morpholog-

ical rules fail to stem proper nouns in most cases, while uncontrolled light stemming removes prefixes in these categories of words, collating the same proper nouns together.

Based on our experience with both collections, neither TREC 2001 nor TREC 2002 could be used alone as a development collection. We suggest that in order to achieve better results, that a development collection be formed from topics in both collections rather than using TREC 2001 as a development collection and TREC 2002 as test collection.

4.3 AIR Experiments on Automatic Speech Recognition Generated Text

Techniques such as normalisation, stopping, tokenisation, root stemming, and light stemming have been shown to increase effectiveness on newswire text [Larkey et al., 2002; Aljlayl and Frieder, 2001]. In this section, we explore a different data set comprising text automatically transcribed from broadcast television news. We use this collection for cross-lingual queries, where queries in English are translated into Arabic, then searched against the collection. This is a very different environment from pure written Arabic monolingual search.

4.3.1 Resources

In this section we describe the collections, translation tools, and stemming algorithms that we use in our experiments.

Collection Description

The TRECVID 2005 data set contains recorded television broadcast news in three languages — Arabic, Chinese, and English — with the associated ASR transcripts available [Over et al., 2006]. Of the total of 169 hours of footage, 43 hours are in Arabic, 52 hours are in Chinese, and 74 hours are in US English. The collection has 24 English-language queries to be used to find specific video footage in the entire collection. The queries all begin with the phrase, "find shots of", and aim to find scenes containing a specific person, place or object, or a general view, building, or action.

The TRECVID ground truth for measurement of retrieval performance is prepared by manually identifying video shots — sections of video footage that correspond to a single camera operation — that satisfy the information need of the user based on the *visual content*.

To create a text-based test set, we aligned the ASR text with the corresponding shots in the video stream. To allow for speed variations and gaps in speech, the text for each shot is

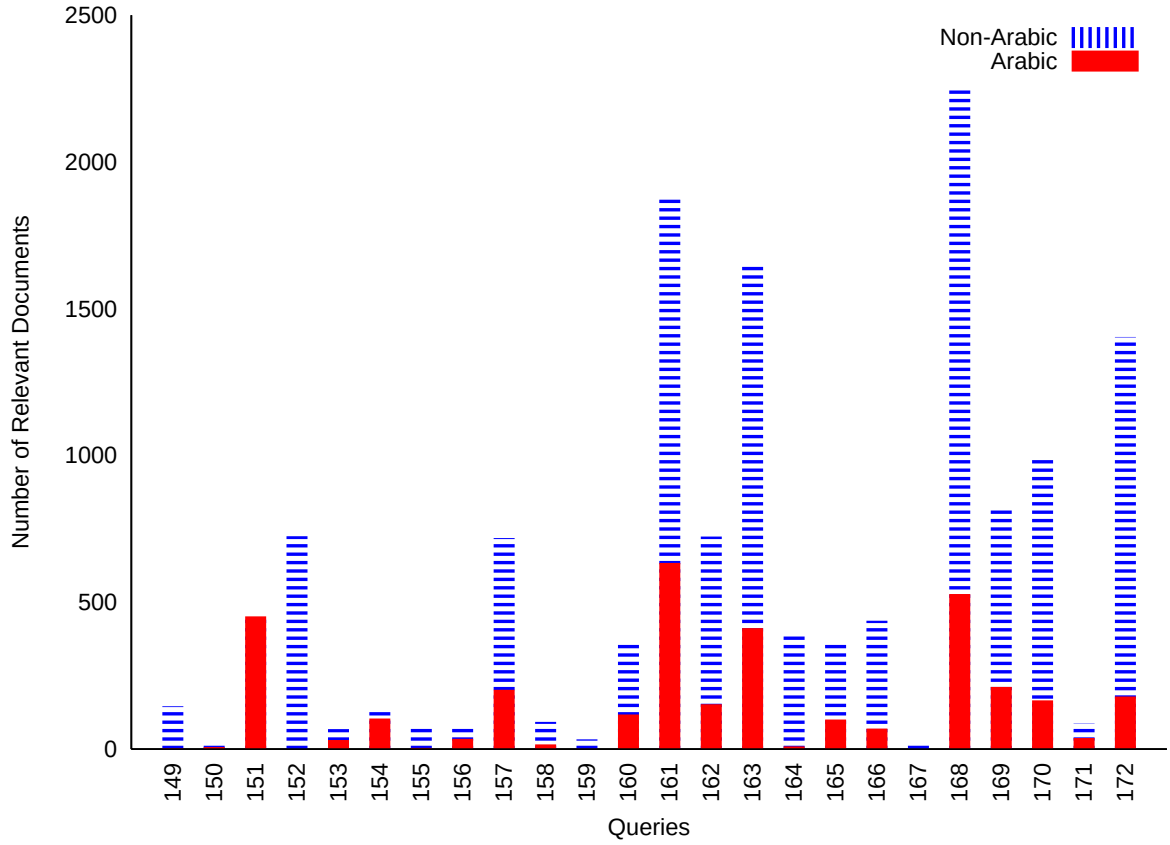

Figure 4.4: The number of documents relevant to each query for the Arabic and non-Arabic documents in the collection

the ASR text that temporally corresponds with that shot and the two shots on either side. The text corresponding to each shot is considered an independent document that is then indexed using a text search engine. A reasonable alternative would be to use story-aligned text [Hsu and Chang, 2005; Hsu et al., 2005], rather than shot-aligned text, as the unit of retrieval; we do not explore story alignment in this work.

We interpret the relevance judgments in the context of this alignment; a document is relevant to the query if its corresponding shot has been indicated as being relevant in the ground truth.

For the work described in this chapter, we focus on only the Arabic data, comprising 26% of the entire TRECVID 2005 collection. The distribution of relevant documents shows that of 13 945 relevant documents, only 3 475 are Arabic. Similarly, the collection-wide average

number of relevant documents per query is 581.0, while for the Arabic subset, the average number of relevant documents per query is 144.8. Figure 4.4 shows the number of relevant Arabic and non-Arabic documents for each query. Naturally, the smaller pool of relevant answers will lead to lower retrieval performance than that reported for work that uses the entire collection. Since we use only the Arabic text, we extracted the relevance judgements for only the Arabic documents in the pool.

4.3.2 Automatic Translation Tools

To evaluate the English queries against the Arabic text, we use three different online automatic translation tools to render the queries into Arabic. These are *AlMisbar*,[4] *Google Translate*,[5] and *Systran*.[6] We expect that the choice of translation tool can affect the quality of the translation, and hence the retrieval effectiveness.

4.3.3 Stemmers and Retrieval Engines

We used the Lemur toolkit to index the collection, and to evaluate the queries against the collection. Lemur incorporates the light10 Arabic stemmer [Larkey et al., 2002] and supports most of the techniques we have described for Arabic search: normalisation, stopword removal, and light stemming.

We have used the Khoja stemmer [Khoja and Garside, 1999] to test the effectiveness of root stemming on this collection. This stemmer removes prefixes and suffixes, and checks for pattern matches after each affix removal; it extracts and returns the root if a match is found in the root-word dictionary, and returns the original word otherwise.

To test tokenisation, where the text in both queries and the collection is split into overlapping tokens of size n and indexed using those tokens instead of complete words, we chose to use n-grams of size three, which have been reported to produce good results for Arabic retrieval [Xu et al., 2002].

4.3.4 Experiments

To evaluate the effectiveness of different techniques used in AIR, we designed five different runs using the translated queries:

[4] http://www.almisbar.com
[5] http://translate.google.com
[6] http://www.systransoft.com

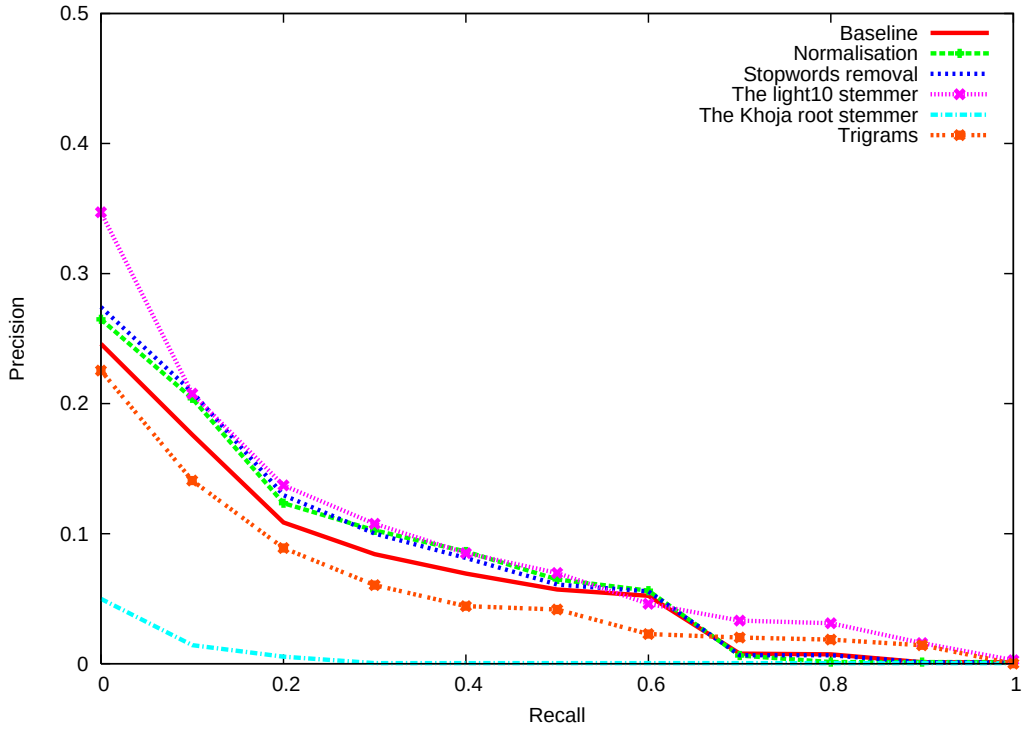

Figure 4.5: Recall-Precision curves for the ASR collection using the different approaches; queries translated with AlMisbar.

1. normalise the queries and run them against the normalised ASR text;
2. stop the queries and run them against the ASR text;
3. stem the queries using the light10 stemmer, and run them against the similarly-stemmed ASR text;
4. stem the queries using the Khoja root stemmer, and run them against the similarly-stemmed ASR text; and
5. tokenise the queries into 3-grams and run them against the similarly-tokenised ASR text.

As a baseline for comparison, we run the translated queries directly against the ASR text.

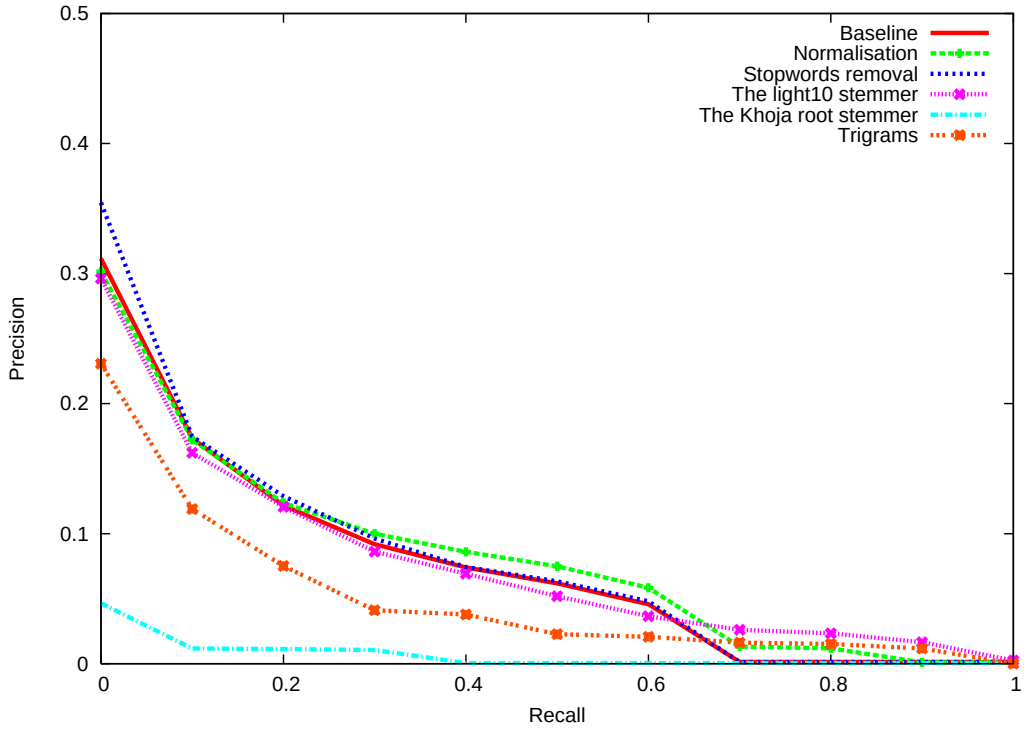

Figure 4.6: Recall-Precision curves for the ASR collection using the different approaches; queries translated with Google Translate.

4.3.5 Results and Discussion

The results for each run are shown in Table 4.24. The techniques have a clear impact on retrieval performance: with the exception of root stemming and trigrams, all produce improved performance over the baseline. As can be seen from Figures 4.5, 4.6, and 4.7, this improvement is consistent across all three translation systems.

This is an important finding, confirming that the approaches are useful even for noisy data such as that used in these experiments. Light stemming appears to produce the most improvement, followed by stopword removal, and then normalisation. Surprisingly, trigrams performed poorer than the baseline.

In contrast to previous reported results [Larkey et al., 2002], root stemming actually leads to poor results on this noisy data.

It is also clear that the choice of machine translation tool has great impact on the results. For instance, the best precision result achieved using Systran is 0.048, which is below the

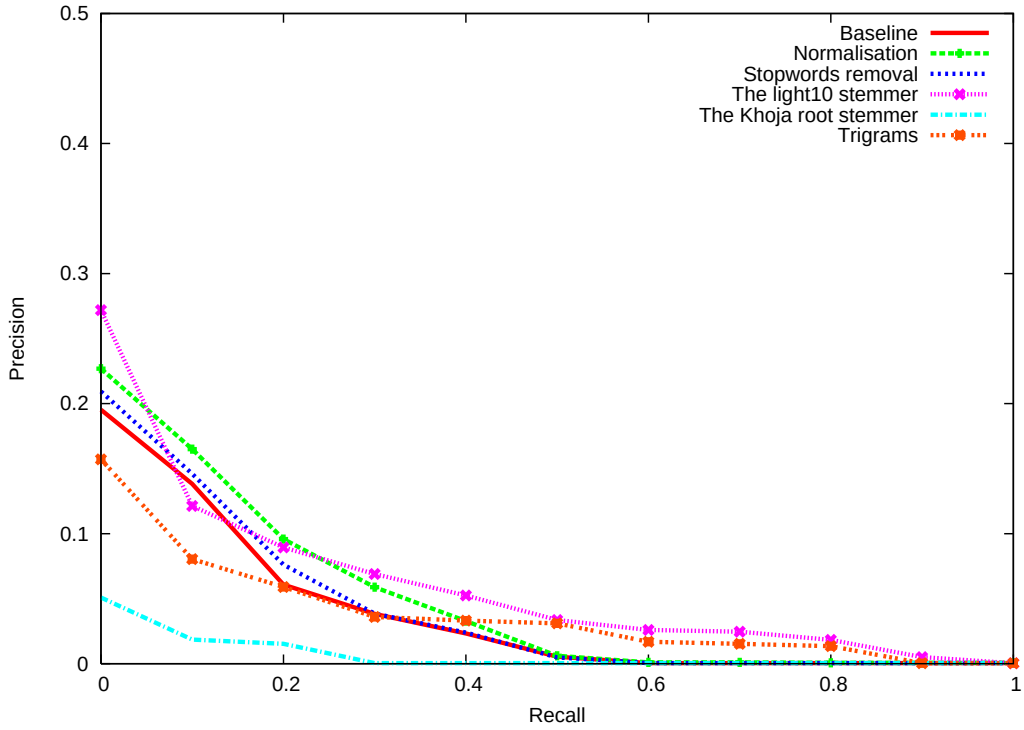

Figure 4.7: Recall-Precision curves for the ASR collection using the different approaches; queries translated with Systran.

baseline result for Google and AlMisbar. Overall, AlMisbar is the best of the three translation systems, producing the highest precision when using light stemming. Figure 4.8 shows the impact of the translation system choice on retrieval performance when applying light stemming.

We observe that root stemming is the only technique that is significantly worse than the baseline when Systran [t-test, $p = 0.049$]. Results produced by the AlMisbar translation system with light stemming are better than the baseline, but the difference is not statistically significant. We note that it is difficult to achieve significant differences based on the relatively small number (24) of available queries.

No automatic translation system is perfect, and, as expected, all three of the translation tools we used had difficulty in finding correct Arabic equivalents for some of the English words in the queries. For instance, the word "court" appears in two English queries, both times in the sense of an open space for games. None of the translation systems produced the correct

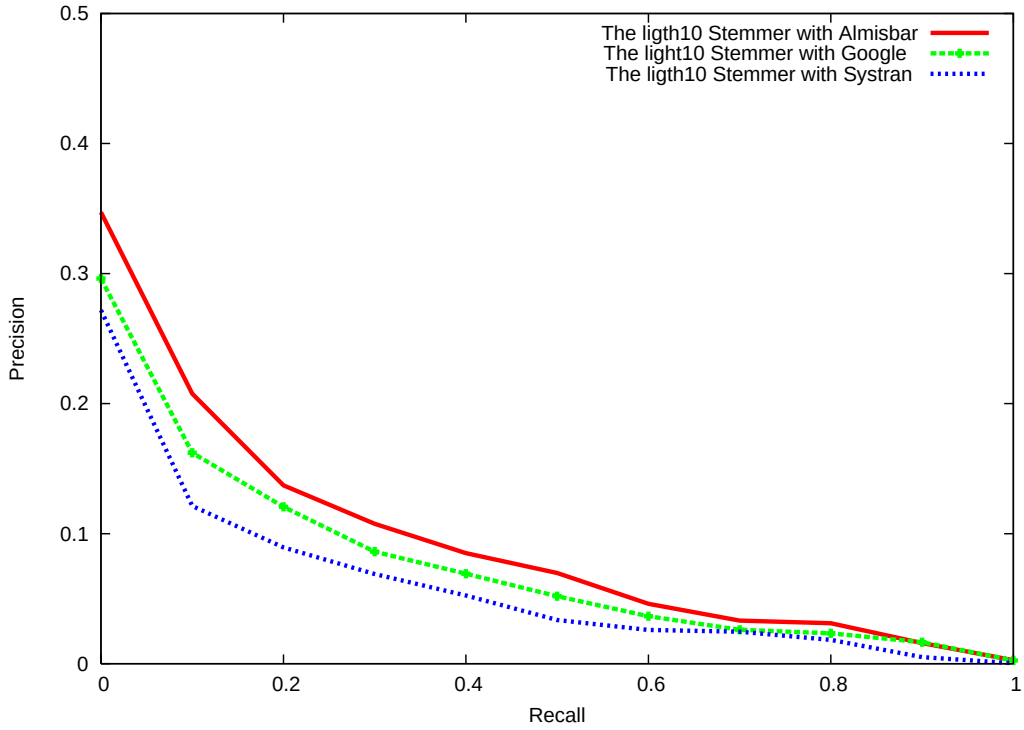

Figure 4.8: Recall-Precision curves for the light10 stemmer across translation systems.

Arabic meaning "مَلعَب" /malʕab/; instead they all translated it to "مَحكَمَة" (/maħkamaa/⟨law court⟩), despite the fact that English queries also contained the word "player".

AlMisbar and Google were both successful in translating most proper nouns in the queries, while Systran transliterated such words inconsistently. Surprisingly, all three translation systems failed to translate the proper noun "Baghdad" to its Arabic equivalent. In addition, Google Translate frequently incorrectly spells words containing the hamza character "ء"; for example, the word "airplane" is translated to the correct meaning but with the incorrect spelling "طَاءرة" /tˤaaʔra/ rather than "طائرة" /tˤaaʔra/.

The noisy data produced by the ASR subsystem is another source of errors, with proper nouns frequently transcribed incorrectly. For instance, the name "Condoleeza Rice" is transcribed completely into one word "كوندوليسارايس" /kwnḍwljsarajs/ instead of two separate ones, and so this this single word is indexed. Since the AlMisbar and Google Translate systems translate the English name into the correct two Arabic equivalents, no match will be found for these terms in the search engine index.

	Machine Translation		
	AlMisbar	Google	Systran
Baseline	0.067	0.069	0.032
Normalisation	0.075	**0.075**	0.041
Stopword removal	0.075	0.071	0.033
light10 Stemmer	**0.081**	0.065	**0.048**
Khoja Root Stemmer	0.003	0.005	0.005
Trigrams	0.053	0.041	0.032

Table 4.24: Effects of different techniques on MAP.

Apart from the use of noisy ASR data and machine translation, our experiments depart from typical information retrieval research in that the underlying relevance assessments are based on the visual content of the shots, and not on the spoken text. Thus, while the comparison of approaches is correct, our absolute results are not directly comparable with other work on Arabic text retrieval. However, the results are comparable to other retrieval results undertaken as part of TRECVID, with the qualification that we use only the Arabic subset of the entire collection of Arabic, Chinese, and English ASR data. Reported results using the full collection of English, Chinese, and Arabic text full collection of English, Chinese, and Arabic text show lower MAP than our results. For example, Foley et al. [2005] reported a MAP of 0.046, and Chang et al. [2005] reported a MAP of 0.039. The MAP that we observed when using the light10 stemmer and AlMisbar translation engine is 0.081.

4.4 Chapter Summary

In this chapter, we have compared the performance of current AIR systems and have shown that the light10 stemmer performs better than other stemmers (except the Buckwalter stemmer when using relevance feedback). We have introduced new stemming techniques that minimise stemming mistakes in light stemming and lead to improved retrieval results. We have used the light10 stemmer as our underlying framework to evaluate techniques that we have developed. We have extended word normalisation for improved retrieval effectiveness, although the difference is not statistically significant. We have also modified automatic generation of stopword variants, with limited success on the three techniques we tried. We have then introduced new techniques to remove single character prefixes: prepositions and

conjunctions. Our empirical results show that these techniques accurately remove prefixes, and so aid retrieval effectiveness. Of the techniques we introduced — RPR, RR, RC, RCL, and RPRRC — RPRRC, in which we remove particles by duplicating the first character and remove the second character if it is a particle by checking the remaining string in the lexicon, performed the best.

We have introduced two new techniques to stem the definite article while avoiding stemming proper nouns. While these techniques are effective, they add no improvement in retrieval performance over the baseline.

We have presented new stemmers for verb prefix removal based on Arabic grammar rules. Our techniques add slight improvements in MAP at the cost of slightly decreased recall. We have also shown that not indexing verbs has a negative effect on retrieval effectiveness.

Using our normalisation approach without joining compound nouns, our conjunction and preposition removal technique RPR, our strict definite article removal technique, our verb prefixes removal technique, and our suffix removal technique with no restriction on stemming some suffixes, we developed the first version of our stemmer "Restrict". When using relevance feedback, this stemmer produces significantly better recall than light10, with an insignificant 3% increase in MAP. Combining our normalisation and prefix removal techniques to create the "Restrict1" stemmer led to a weakly significant improvement in MAP over the baseline when using relevance feedback.

We have demonstrated how to remove suffixes based on Arabic grammar rules; overall, suffix removal did not improve retrieval effectiveness over the baseline. Combining our prefix removal and suffixes removal techniques in the form of the "Restrict2" stemmer improved retrieval performance over the baseline, but the improvement was not statistically significant.

We have developed three improved versions of the light10 stemmer that we refer to as light11, light12, and light13. In the light11 stemmer, we have reduced the number of suffixes and stopwords removed by the original light10 stemmer. In the light12 stemmer, we reduced the list of suffixes to be stemmed to four suffixes. In light13, we remove the same suffixes as light12, and also remove the definite article and prepositions and conjunctions using our rules. We have empirically found that while light11 and light12 do not significantly improve retrieval performance, light13 exhibits significantly better precision than the baseline in conjunction with relevance feedback.

To maximise the portability and the efficiency of our system, we extracted all possible words with seven or fewer characters from the lexicon used in our experiments, and used it instead of the Microsoft Office 2003 lexicon. The extracted lexicon is incomplete, as it does

not include words with eight or more characters. This resulted in reduced performance.

Using the unique terms of the collection instead of using the Microsoft Office 2003 lexicon to validate stemming operations resulted in improved performance, with two of our stemmers performing better than the baseline, and significantly better when using relevance feedback.

We ended the chapter by evaluating the effect of several pre-processing and translation approaches for a noisy data set of Arabic text. Our results show that stopping, light stemming, and tokenisation improve retrieval effectiveness, but that root stemming and trigrams have a negative impact. We have also shown that the choice of the machine translation engine has a large impact on measured performance in such experiments.

In the next chapter, we explore the effects of using a larger text collection on the AIR stemmers we have reviewed here.

Chapter 5

Corpus Size Effects on AIR Systems

Test collections play a core role in improving IR systems, as they allow different strategies to be tested. For Arabic, the few available test collections are small compared to those used for English. For example, while the largest test collections developed for Arabic — the TREC 2001 and TREC 2002 text collections — contain only 383 872 documents (some 800MB of data), the English TREC WT10g collection contains 1.6 million documents (10GB of data), and the English TREC GOV2 text collection contains 25 million documents (420GB of data).

There are only 25 queries in the TREC 2001 collection and 50 queries in the TREC 2002 collection. As pointed out in Section 2.3.2, the TREC 2001 topics have been reported to lead to unreliable results due to forming the pool with a large number of documents retrieved by only one participant system. Our results in the previous chapter show that the two collections often give incompatible or contradictory results. We have found that using the TREC 2001 collection as a training collection leads to far better improvements than using the TREC 2002 collection.

Our intention is to test techniques to find variants of foreign words within an IR context. The TREC 2001 and TREC 2002 queries have only 15 foreign words in total, which is not sufficient to test algorithms developed to unify different forms of foreign words.

In this chapter we use a larger text collection, and build a larger set of queries with associated relevance judgements, with the aim of testing the effects of larger corpus size on AIR systems, and the effects of foreign words in Arabic queries.

5.1 Building a Test Collection

As described in Section 2.3.1, in order to build a test collection, three components are required, namely a document collection, queries and corresponding relevance judgements. In the following subsections we describe the document collection, the task, participants, and the annotation process.

5.1.1 The Document Collection

There are several options available to obtain a larg text collection. These include crawling Arabic pages from the Web, or using available text collections.

The first option requires much time and effort to build the document collection, the topics, and the relevance judgements. To produce a homogenous text collection from the Web, Arabic web pages need to be crawled, different encodings need to be unified, and noisy unrelated links and text need to be removed. Since creating a collection is not central to this thesis, we adopted the second approach, where we use large text collections prepared by IR or NLP specialists. Such text collections are usually collected from news agency dispatches over a period of time. The TREC 2001 and TREC 2002 collections are examples of such corpora.

The Arabic Gigaword Document Collection

We settled on the second edition of the Arabic Gigaword collection [Graff et al., 2006] produced by the Linguistic Data Consortium (LDC).[1] This contains documents acquired between 1994 and 2004 from several news agencies: Agence France Presse, Al-Hayat, An-Nahar, Ummah Press, and Xinhua.

The collection has 1 591 987 documents in over 5GB of uncompressed data. Documents are tagged using SGML tags similar to the TREC 2001 collection described in Section 2.3.2. Two additional tags are "type" and "dateline". The "type" tag classifies a document as a "story", "multi", or "other". A document that is categorised as a "story" describes a coherent report on a particular topic or event, a document that is categorised as "multi" describes unrelated events such as news headlines, and a document that is categorised as "other" is a document that does not fall in the previous two categories. Each document is assigned a unique ID, a language code (in our case, "ARB", as all the documents are in Arabic),

[1] http://www.ldc.upenn.edu/

a date, and a four-digit sequence number for that date. For example, the document ID "AFP_ARB_20020522.0097" implies that this document is from the AFP newswire, written in Arabic on 22nd May 2002, and is document number 97 in that collection for that date.

We have found some 344 documents that do not completely follow this labelling convention — but this does not affect our experiments, and we did not modify them.

As this document collection does not have any queries or relevance judgements, we have decided to build such queries and their relevance judgements using native Arabic speakers.

5.1.2 The Task

To help Arabic native speakers in creating topics and the relevance judgements, we prepared a document in Arabic explaining the task and the objectives of our experiment. We described the content of the AGW corpus and the task required of the participants in simple language, and exemplified the task as using the Google search engine. We asked the participants to perform two main tasks:

- Come up with a list of queries.
- Search our news collection and annotate relevant documents to their queries.

Each query has to include: a title, a small statement that the user usually types in the search engine; a description, a longer statement that explains the user information needs; and a narrative field, that details exactly what is related and not related to the user query.

Since the documents all date between 1994 and 2004, we provided participants with a list of events from that period translated from Wikipedia year reports.[2] Foreign proper nouns were left untransliterated so that participants would themselves transliterate any words they needed. We also included the original English description of the event. The Arabic and English summaries for each year are available online at:
http://www.cs.rmit.edu.au/~nwesri/Research/Events.

Twenty participants, all male, were involved in the annotation process. They were all university students. At the time of the experiment, seven of them held a master degree and were pursuing PhD degrees, while the rest held undergraduate degrees in science and engineering. Two of the participants were from Saudi Arabia, one was from Iraq and the rest were Libyans.

[2] http://en.wikipedia.org/wiki/YEAR. Replace the "YEAR" with 1994 to 2004.

Figure 5.1: Our AGW annotation system. Users can enter new versions of their query and mark relevant documents.

5.1.3 Annotation System

Every participant was given a list of events that happened in a particular year and was asked to provide at least seven topics of his interest. At least three of these topics had to include foreign terms. Examples from the TREC 2001 topics were given. Participants used an online system to read the guidelines and documents. The topic definition stage produced a total of 122 topics; we removed duplicates except in cases where the information need was different.

5.1.4 Annotation Methodology

We choose to use the ISJ technique described in Section 2.3.1 to create the relevance judgements. We decided to use this technique for the following reasons:

- Participants are all typical internet users and are are familiar with the search process.
- The lack of different techniques that consider foreign words in retrieval. We believe that using pooling with the existing AIR systems would fail to capture variants of foreign terms in our queries within the entire collection.
- To save time and effort for unpaid annotators.
- ISJ allows the users to modify their queries and annotate relevant documents that in some cases do not include any terms they initially wrote in their queries but satisfy their needs. This is also good in our case as participants are made aware of the problem of foreign word variants and Arabic word variations.

We implemented a web-based system that allows users to search the AGW collection. The AGW text collection was indexed using the Indri search engine, which supports indexing text encoded in the UTF-8 format. We developed a user interface using the API application provided with the Lemur toolkit. All participants were given an account from which they were able to view their topics and search the document collection. Initially the collection is searched with the original version of the query, with the ability to modify the query and search the collection as many times as the participant wants. Participants can view the documents and mark those that they consider to be relevant. The system provides some visual cues to help the user identify whether a document is relevant or not. These include colouring the keywords in the query within the document that the user is viewing. Different colours are used for each term in the query that appears in both the query and the document. Participants reported that this feature was very helpful. Figure 5.1 shows a screenshot of the annotation page of our system. The system retrieves documents with keywords matching the query, and lets the user annotate them. To increase the efficiency and minimise the time of loading web pages while the annotation is in progress, the system prefetches the top hundred retrieved documents from the server to the user's computer. For every annotation, we store the document ID, the topic number, the topic version that the user types, and the timestamp of that transaction.

Once the user saves his work, the system fetches the next hundred unannotated documents, excluding the already annotated documents for that particular topic. To facilitate using variants of foreign words, the system is fed with possible variants for all foreign words found in the list of events, plus topics written by the users. Variants are determined using using the top five unique words returned by the ten approximate string- and phonetic-matching

Query Variant	Not Relevant	Relevant	Total
اعتقال صدام حسين	7	1	8
يوم اعتقال صدام حسين	26	2	28
القبض على صدام حسين	54	8	62
نهاية صدام حسين	25	2	27
فرار صدام حسين	81	0	81
صدام حسين اسر	25	0	25
صدام حسين قبو	26	17	43
صدام حسين الدوار	10	2	12
حفره صدام حسين	53	27	80
صدام حسين مختبئا	23	2	25
اعتقلت صدام حسين	52	10	62
كيفية اعتقال صدام حسين	91	0	91
كيفيةاعتقال صدام حسين	81	9	90
Total annotations	554	80	634

Table 5.1: Variants of topic number 13 "إعتقال صدام حسين" ⟨The capture of Saddam Hussein⟩; entered by a user to annotate relevant documents. Most relevant documents are captured when typing the word "حُفرة" (/ħufra/⟨a hole⟩) and the word "قَبُو" (/qabuw/⟨Cellar⟩). Both words are not found in the original query.

techniques described in Section 7.2, and indexed using the version of foreign words found in events and topics. If the user selects a topic with a particular foreign word, the system displays the list of possible variants at the top and the user is free to use any version. Not all the variants will necessarily be correct, nor do we manually refine them.

Annotators were also told to pick any terms that they think will improve retrieval from the documents that they view. They were allowed to update the description and the narrative fields, but not the title field. In practice, none did so, except to correct spelling mistakes.

Users were made aware of the effects of Arabic word variants on retrieval results. We explained the way search engines search Arabic text, and showed the difference between searching the document using the different versions of Arabic words and the effects of Arabic affixes on retrieval. For example, we told participants that a word with the definite article in the query might return different documents than the same word without the definite article.

5.1.5 Annotations

Over six 2-hour sessions, the participants were able to annotate more than 26 000 documents for a total of 122 topics. In the first session, we explained the systems to participants, and demonstrated how to search the collection. They were also briefed about possible variations of Arabic and foreign words. They initially searched the collection using their own topics that we had fed to the system. Every participant could view his own topics, select, and search them in the AGW collection. In about two hours, participants were able to annotate over 3 000 documents. In the second session, we normalised the text using techniques described in Section 2.2.1. By the end of this session, the total number of annotated documents was around 7 000. In session three, we automatically expanded queries using variants of foreign words identified automatically as described in the previous subsection. We included all variants as synonyms to the original foreign words in the query. This technique was not successful as participants were not satisfied with the retrieval results and so, we stopped this in the following sessions and only maintained the index where text is normalised. By the end of session three, the number of annotations reached 12 915, and by the sixth session, we had collected 26 493 annotations. Out of the 122 topics submitted we decided to use only 90 topics and their relevant annotations. We dropped topics that had no relevant documents in the collection or where all the annotated documents were relevant. We also dropped topics with fewer than 100 annotations. Several similar topics, annotated by different annotators, were retained due to the fact that their information needs and the overlap in the annotated relevant documents are different. For example, queries 20 and 65, and queries 24 and 76 are almost identical; however they differ in the number of keywords, the number of annotated documents, and the number of relevant documents. For example, query 20 has 7 keywords while query 65 has only 3. Both queries have a similar number of annotations, with a similar relevance ratio, but while the overlap between the total annotations from both queries is 75 documents, only 4 documents are annotated as relevant to both queries.

The final 90 topics have 25 782 annotations with 4 036 relevant documents. The topics and their respective statistics are shown in Appendix A. Table 5.1 shows an example of query variants typed by a participant and their corresponding annotations.

The final relevance judgements are formulated in the 4-column TREC format (see Table 2.5 in Section 2.3.1). This format allows us to use tools such as `trec_eval`[3] to easily compute precision and recall measures.

[3] http://trec.nist.gov/trec_eval/

Technique	MAP	P@10	R-Precision	Recall
noStemming	0.1414	0.2267	0.1676	0.5572
light10	0.1747	0.2567	0.2015	0.6080
B.Stem	**0.1795**	**0.2656**	0.2103	0.6162
B.Stem2	0.1684	0.2511	0.2005	0.5864
B.Lemma	0.1712	0.2622	0.2004	**0.6305**
Al-StemN	0.1785	0.2622	**0.2117**	0.6181
Al-Stem	0.1740	0.2567	0.2011	0.6139
Khoja	0.1491	0.2189	0.1847	0.5594
noStemming	0.1521	0.2300	0.1789	0.6300
light10	**0.2063**	**0.2878**	0.2214	0.7009
B.Stem	0.2026	**0.2878**	**0.2316**	0.7175
B.Stem2	0.1910	0.2744	0.2140	0.6959
B.Lemma	0.1822	0.2589	0.2064	0.7232
Al-Stem	0.1960	0.2767	0.2186	0.7118
Al-StemN	0.1947	**0.2878**	0.2192	**0.7244**
Khoja	0.1561	0.2356	0.1790	0.6315

Table 5.2: Performance of existing Arabic stemmers using the AGW test collection. The top half of the table shows results without using relevance feedback, while the bottom half shows results with relevance feedback.

5.2 Performance of AIR Stemmers on The AGW Test Collection

In this section, we report results of running the AIR stemmers used in the previous chapter on the new test collection. We first show the results of running existing AIR stemmers, then we run our new stemmers and compare them with the baseline. We use the same experimental settings described in Section 4.1.2, and show results with and without query expansion using pseudo relevance feedback.

5.2.1 Performance of Existing AIR Stemmers Using The AGW Test Collection

Table 5.2 shows results of running the Larkey light10 stemmer, the Buckwalter stemmers, the Al-Stem stemmers, and the Khoja root stemmer.

All AIR stemmers — except the Khoja stemmer — add significant improvement in both

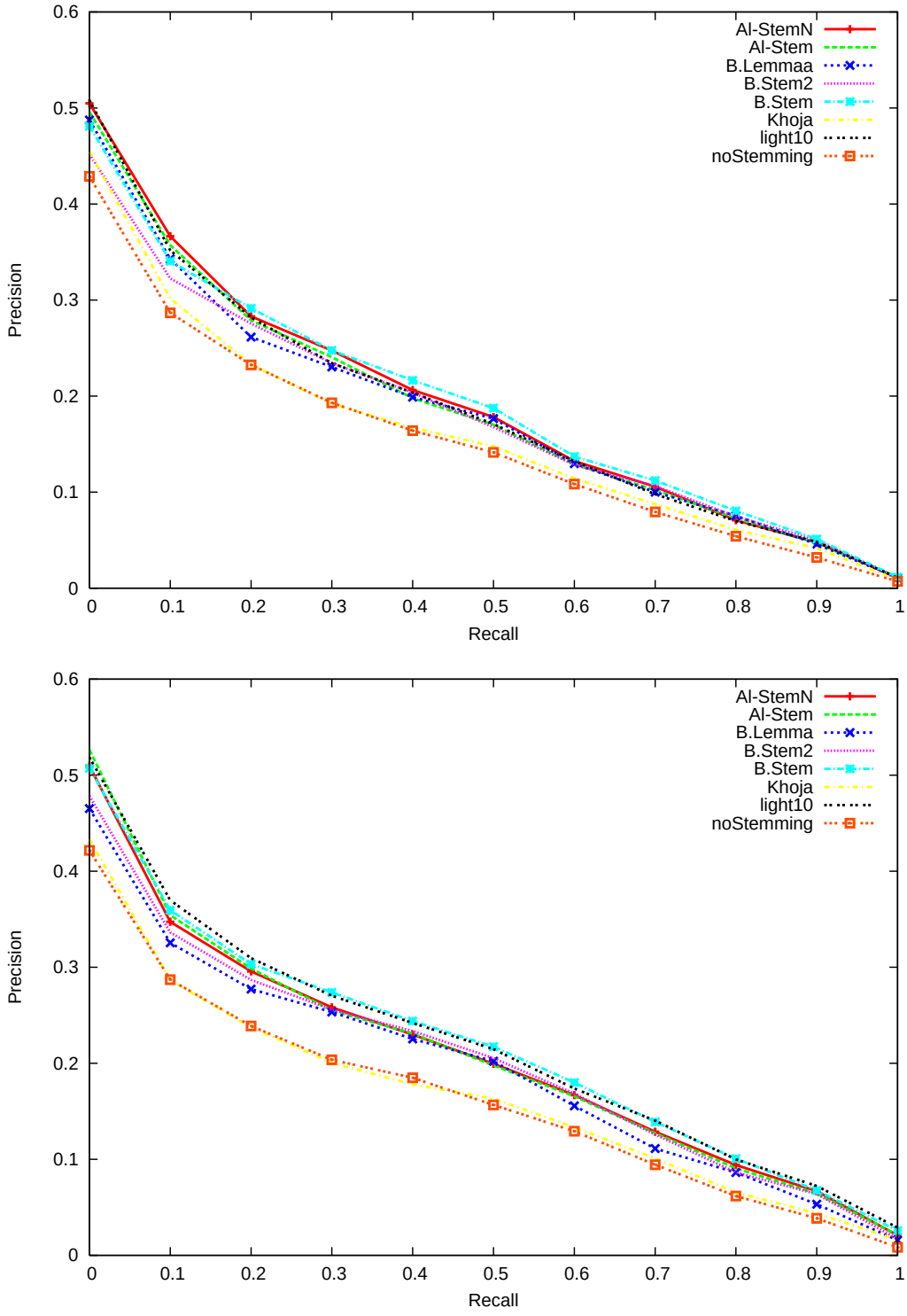

Figure 5.2: Performance of AIR stemmers using the AGW test collection. Performance without using relevance feedback (top), and with relevance feedback (bottom).

MAP, and R-Precision to the noStemming approach. Only the B.Stem and B.Lemma add significant improvement in P@10 [t-test, $p = 0.045$, and $p = 0.056$ respectively]. No stemmer is significantly distinguished in terms of recall.

All stemmers except B.Lemma and B.Stem2, outperform the Khoja root stemmer significantly. The B.stem, and Al-StemN stemmers are significantly better at the 99% confidence level [t-test, $p = 0.007$, and $p = 0.009$ respectively]; and Al-Stem and light10 stemmers are significantly better at the 95% confidence level [t-test, $p = 0.028$, and $p = 0.011$ respectively].

The Larkey light10 stemmer performs poorer than the B.Stem or the Al-StemN, but the difference is not statistically significant.

Among the Buckwalter stemmers, the B.Stem approach, where the first returned stem is used, is the best. Despite the fact that the stemmer has the second best MAP, it is significantly better only than the B.Stem2 and the Khoja stemmers [t-test, $p = 0.027$, and $p = 0.007$ respectively].

Al-StemN shows better performance than the original Al-Stem. This is due to the effects of removing stopwords and non-letters. The improvement in MAP and R-Precision is statistically significant [t-test, $p = 0.007$, and $p = 0.004$ respectively]. The top half of Figure 5.2 shows the performance of these AIR stemmers over the 11 standard recall points.

To show the effects of query expansion using pseudo relevance feedback on retrieval, we used the same parameters as in the previous chapter and set the relevance feedback parameters in the Lemur toolkit to use the top 20 terms returned in the top 15 retrieved documents to expand queries. Overall results improved significantly. Table 5.2 shows improvements over the original stemmers.

All stemmers improve with relevance feedback; notably, it significantly increases the MAP of the light10 [t-test, $p = 0.004$], B.Stem [t-test, $p = 0.027$], B.Stem2 [t-test, $p = 0.022$], and Al-Stem [t-test, $p = 0.045$] stemmers.

Stemmers also show a significant improvement over the noStemming and the Khoja root stemmer except the B.Lemma, which is better than the Khoja stemmer only at the 93% confidence level [t-test, $p = 0.071$].

The light10, and the Buckwalter stemmers perform the best. There are no other significant differences between the performance of these stemmers on this collection.

In terms of recall, the best stemmers are Al-StemN and B.Lemma. The Al-StemN approach is significantly better than B.Stem2 and B.Lemma is not significantly better than other stemmers. The bottom half of Figure 5.2 shows the performance of AIR stemmers when using relevance feedback.

Technique	MAP	P@10	R-Precision	Recall
light10	0.1747	0.2567	0.2015	0.6080
B.Stem	0.1795	0.2656	0.2103	0.6162
Restrict	0.1752	0.2644	0.2017	0.6015
Restrictc	0.1828	0.2678	0.2122	0.6214
Restrict1	0.1805	**0.2722**	0.2114	0.6105
Restrict1c	**0.1830**	0.2578	**0.2131**	0.6204
Restrict2	0.1739	0.2611	0.2025	0.6258
Restrict2c	0.1825	0.2567	0.2107	0.6263
light11	0.1803	0.2622	0.2059	0.6100
light12	0.1785	0.2600	0.2008	0.6276
light13	0.1820	0.2556	0.2087	**0.6305**

Table 5.3: Results of our new stemmers compared with the light10 stemmer and the B.Stem stemmer. Our stemmers show better performance over the two stemmers, although the difference is not significant.

5.2.2 Performance of our Stemmers on The AGW Test Collection

In this section we compare the results of our stemmers with the light10 stemmer and the B.Stem stemmer. To test our Restrictc, Restrict1c, and Restrict2c algorithms, we formed our lexicon using the unique words in the AGW corpus.

In a similar way, we have run our new stemmers with and without relevance feedback. Results of running stemmers without relevance feedback are shown in Table 5.3, and results of running stemmers with relevance feedback are shown in Table 5.4

Restrictc, Restrict1c, and Restrict2c outperform both the light10 and B.Stem stemmers. Restrictc has a weakly significant improvement in MAP over the light10 stemmer [t-test, $p = 0.064$]. It is clear that our stemmers have a better recall rate than the baseline stemmers. Among our new light stemmers, the light12 stemmer performs below B.Stem, but not significantly so. The light13 stemmer shows higher recall than all stemmers, but the P@10 is lower. Overall, all stemmers perform almost equally with no major change in any of the measures.

With relevance feedback, however, both B.Stem and the light10 show better performance than all other stemmers. The recall rate of the light10 stemmer is significantly better than

Technique	MAP	P@10	R-Precision	Recall
light10	**0.2063**	**0.2878**	0.2214	0.7009
B.Stem	0.2026	**0.2878**	**0.2316**	**0.7175**
Restrict	0.1926	0.2744	0.2142	0.7004
Restrictc	0.2047	0.2789	0.2238	0.6888
Restrict1	0.1962	0.2822	0.2136	0.6684↓
Restrict1c	0.1988	0.2622	0.2217	0.6764
Restrict2	0.1865	0.2678	0.2065	0.6989↓
Restrict2c	0.1984	0.2856	0.2194	0.6724
light11	0.2034	0.2778	0.2215	0.7001
light12	0.1980	0.2833	0.2158	0.7036
light13	0.1987	0.2700	0.2193	0.7031

Table 5.4: Results of running our new stemmers using pseudo relevance feedback. None of our stemmers show better performance than the light10 stemmer or the B.Stem stemmer. ↓ shows values that are significantly worse than the B.Stem

Restrict1 [t-test, $p = 0.040$] and Restrict2 [t-test, $p = 0.040$], and weakly significantly better than Restrict2c [t-test, $p = 0.085$]. B.Stem has significantly better recall than Restrict2 [t-test, $p = 0.045$]. The stemmer is also weakly significantly better in terms of R-Precision [t-test, $p = 0.072$]. Our light stemmers has similar recall to the light10 stemmer.

Compared with other stemmers, Al-StemN has significantly better recall than Restrict1, Restrict2, and Restrict2c [t-test, $p = 0.038$, $p = 0.023$, and $p = 0.039$ respectively]. In terms of other measures, no significant changes are seen by any other stemmers.

By looking at the number of terms, the number of unique terms, and the index size produced by each algorithm (presented in Table 5.5), our algorithms that use rules to stem the collection produce the lowest number of unique terms, indicating that they conflate words better than the other algorithms. The numbers also show that using the corpus as an underlying lexicon in our stemmers conflates words better than using the Microsoft Offices 2003 lexicon (Restrict1 verses Restrict1c). This improves not only retrieval effectiveness but also efficiency, as the number of index entries is reduced. Our corpus-based-lexicon algorithms stem the AGW collection in about 68 minutes compared to over 700 minutes using the same algorithms with the Office 2003 lexicon, over 500 minutes using the Buckwalter stemmer, and 50 minutes using the light10 stemmer.

	Index		
	Terms	Unique terms	Size (Kilobytes)
Restrict1c	373 024 424	751 975	3 074 176
B.Lemma	371 689 003	757 859	3 002 652
Restrict1	372 997 651	770 496	3 066 112
Khoja	376 957 637	775 526	2 941 852
Restrictc	372 935 539	822 554	3 077 384
Restrict2c	372 806 110	843 417	3 079 320
Al-StemN	359 212 138	864 300	2 986 332
light13	373 044 083	875 193	3 115 972
light10	375 495 701	920 744	3 130 328
Restrict	374 437 914	934 763	3 102 336
B.Stem	371 996 493	957 215	3 040 956
light11	373 421 304	994 558	3 132 716
Restrict2	372 814 421	989 199	3 083 468
Al-Stem	431 337 555	1 003 733	3 536 860
light12	373 050 416	1 025 464	3 144 592
B.Stem2	373 127 231	1 034 282	3 099 628
noStemming	479 010 563	1 956 315	4 202 384

Table 5.5: Number of terms, unique terms, and index size of each algorithm. Algorithms that use rules produce fewer unique terms than those that do not use rules in stemming. Stemmers are ordered according to the number of unique terms they return.

The difference in the number of terms returned by the different algorithms is due to the difference in parsing strategies. In our stemmers, we split words that are joined by mistake, join compound nouns, and remove the one-character terms from the stemmed text. This results in different term numbers. The fewest terms are produced by Al-StemN, possibly due to the uncontrolled affix removal followed by the stemmer. The stemmer does not imply a limit on the remaining stems and removes a large number of prefixes and suffixes, resulting in many zero-length stems.

We expected that the Khoja stemmer would produce the lowest number of unique terms as it conflates words to roots. However, this is not the case when using the AGW collection. The stemmer produces more unique terms than the Restrict1, B.Lemma, and Restrict1c

stemmers. However, the index size of this stemmer is the smallest as most of the returned stems are roots with three characters, whereas the other stemmers do not process words to the three-letter stem.

5.3 Discussion

Results with the new data set — with more queries and a larger collection — show that our stemmers work better than other stemmers when no relevance feedback is used. But this is not the case when using relevance feedback.

As our stemmers use the unique terms in the collection as an underlying lexicon, they will not achieve optimal performance if the lexicon is not comprehensive, or if it includes misspelled words. When using the AGW corpus unique terms as a lexicon, we discovered many misspelled words in the AGW corpus, making it problematic to use unique terms from this corpus for the lexicon. For example, the technique Strict *al* (SAL), described in Section 4.9, stems the prefix "الـ" only if adding another *al* results in incorrect word. This technique fails to remove the definite article from the word "الشريف" (/alʃrjf/⟨the honorable⟩) due to the fact that the word "الالشريف" /alalʃrjf/(misspelled form of word "الشريف") is found within the corpus terms. Another example is stemming the proper noun "حسين" (/ħsjn/⟨Husain⟩) to "حِس" (/ħis/⟨feeling⟩) as replacing the last two letters "ـين" with "ـون" yields the proper noun "حسون" /ħswn/ and replacing them with "ـان" produces another proper noun "حسان" /ħsan/ which satisfies rule 4.10 (described in Section 4.13).

Results also suggest that simple light stemming approaches are still an effective option to increase the retrieval effectiveness — regardless of stemming mistakes — as they are more efficient and produce similar results. With the light12 stemmer, we have demonstrated that stemming only four suffixes and seven prefixes, without any rules, is as effective as all other rule-based algorithms, where extensive word analysis is carried out.

Our results show that improvement in search effectiveness over the baseline with large collections does not mirror those reported using the relatively small TREC 2001 and TREC 2002 collections. Larkey et al. [2007] reported a 100% improvement, and Aljlayl and Frieder [2002] reported an improvement of 87.7%. However, with the AGW collection, we find that the improvement for the light10 stemmer is only 23.6%. We believe that this is due to the difference in the sizes of the collections, the number of queries, and the nature of words in the queries. More than half of the AGW queries contain foreign terms that require special attention compared to normal stemming. The performance of the light10 stemmer on the

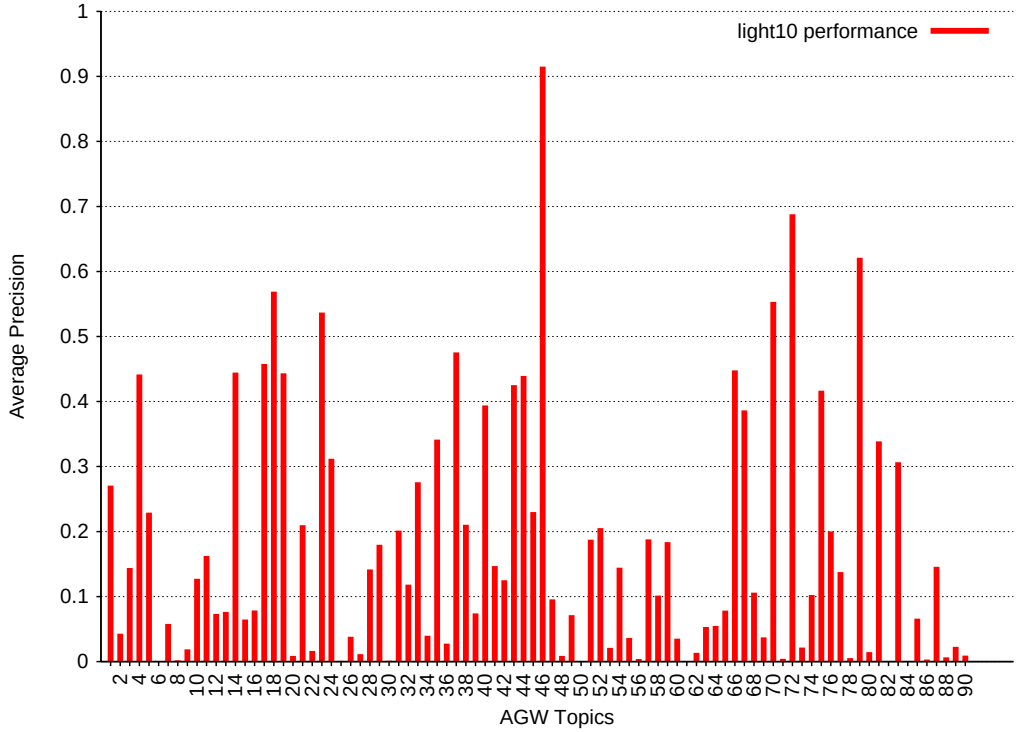

Figure 5.3: The performance of the light10 stemmer on the individual queries. 57 queries score less than 0.2 in average precision (AP).

individual queries shows that 60 queries score less than 0.2 in AP, including 43 queries that score less than 0.1; this is shown in Figure 5.3.

Finally, we find that root stemming produces the poorest retrieval results on the large AGW collection. Using a much smaller collection, [Al-Kharashi, 1991] showed that root stemming produced better results than light stemming. On mid-sized collections, root stemming has been both shown to be almost equal [Larkey et al., 2002] and worse than light stemming [Aljlayl and Frieder, 2002; Darwish and Oard, 2003a]. We conclude that this approach is sensitive to corpus size.

5.4 Tuning Okapi BM25 Ranking Parameters

The Okapi BM25 ranking function (see Equation 2.6) has three adjustable parameters: b, k_1 and k_3. As discussed in 2.6, the b parameter is used to normalise the document length and takes values between 0 and 1, where 0 specifies that no document length normalisation be

performed, and 1 specifies full normalisation. The parameter k_1 affects the term weight in the document, If it is set to 0, then the term weight is not affected by its frequency in the document. The third parameter, k_3, is related to the term frequency in the query. If it is set to 0, then only one instance of the term in the query is used in the ranking.

The optimal values of these parameters for English collections were determined in the TREC 8 ad hoc experiments by Robertson and Walker [1999], who recommended that b be set to 0.75, k_1 be set to 1.2, and k_3 be set to values between 7 to 1 000 for long queries. These values are the default values in the Okapi retrieval model in the Lemur toolkit. El-Khair [2003] reported that he could not determine any betters values for Arabic using the TREC 2001 collection. We use these parameters and check the effects of using other values on retrieval using our new collection.

To determine the best value for the different parameters, we used the result of running the light11 stemmer on both the queries and the collection using the TREC default values as our baseline. We use the light11 stemmer as it was found to be effective in Chapter 4.

5.4.1 The b Parameter Value

To determine the best value for document length normalisation in the AGW collection (b), we test all values from 0 to 1 with an interval of 0.05. Each time we change the b value, we run queries against the collection index and record results. Figure 5.4 shows the effects of changing the value of this parameter, and results are shown in Table 5.6.

It is clear that when b is set to 0.0 or values that are greater than the default value, the performance of the light11 stemmer decreases significantly. Values in the range 0.05 to 0.7 increase the MAP measure significantly with a confidence level of 99%. The same values have similar effects on the R-Precision measure except for the values 0.05, 0.1, 0.15 and 0.7, which result in insignificant improvements; and the value 0.65, which improved the light11 significantly but only at the 95% confidence level [t-test, $p = 0.030$]. The same range of values has also affected the P@10 measure. Values 0.15 and 0.2 have also significantly improved performance but at the 95% confidence level [t-test, $p = 0.022$, and $p = 0.010$ respectively].

The best performance is seen when the parameter b is set to 0.25. This value produces the best values in MAP, P@10, R-Precision and Recall. It improves MAP, P@10, and R-Precision at the 99% confidence level [t-test, $p < 0.01$], and improves recall, although not significantly so at the 99% nor the 95% confidence levels [t-test, $p = 0.090$].

Value of b	MAP	P@10	R-Precision	Recall
0.0	0.1770↓	0.2689	0.2047	0.6278
0.05	0.1886⇑	0.2833⇑	0.2162	0.6397
0.1	0.1971⇑	0.2900	0.2216	0.6449
0.15	0.2010⇑	0.2956↑	0.2262	0.6484
0.2	0.2045⇑	0.2967↑	0.2324⇑	0.6484
0.25	**0.2051**⇑	**0.2978**⇑	**0.2341**⇑	**0.6499**
0.3	0.2048⇑	**0.2978**⇑	0.2301⇑	0.6486
0.35	0.2047⇑	0.2933⇑	0.2300⇑	0.6446
0.4	0.2022⇑	0.2911⇑	0.2273⇑	0.6402
0.45	0.2005⇑	0.2911⇑	0.2259⇑	0.6375
0.5	0.1988⇑	0.2900⇑	0.2254⇑	0.6357
0.55	0.1959⇑	0.2922⇑	0.2210⇑	0.6318
0.6	0.1924⇑	0.2900⇑	0.2216⇑	0.6290
0.65	0.1884⇑	0.2833⇑	0.2124↑	0.6233
0.7	0.1844⇑	0.2800⇑	0.2113	0.6174
0.75	0.1803	0.2622	0.2059	0.6100
0.8	0.1742↓	0.2556	0.2016	0.6057
0.85	0.1670↓	0.2378↓	0.1957↓	0.5976
0.9	0.1598↓	0.2211↓	0.1904↓	0.5874↓
0.95	0.1539↓	0.2133↓	0.1818↓	0.5805↓
1.0	0.1468↓	0.1978↓	0.1700↓	0.5730↓

Table 5.6: The results of changing the Okapi b parameter. Values range from 0 to 1 at constant increase with 0.05. ↑ indicates values that are significantly better than the default value (0.75) at the 95% confidence level, ⇑ indicates those significantly better at the 99% confidence level, while ↓ indicates values the are significantly worse than the default value.

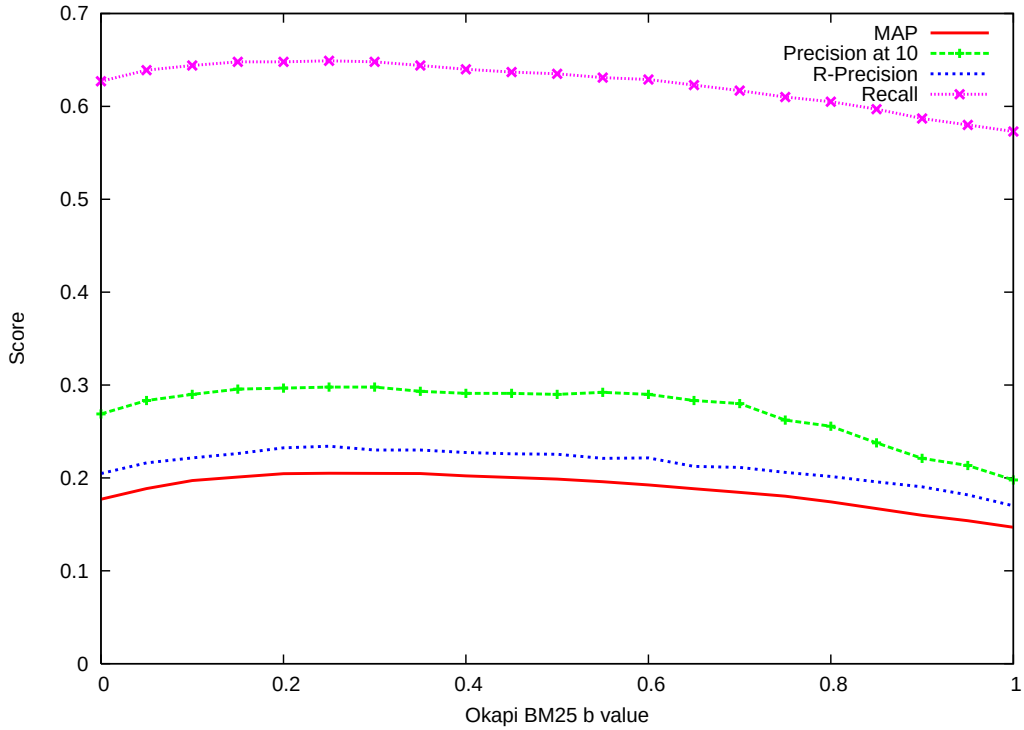

Figure 5.4: The effects of changing the b value while fixing the other two parameters to their default values. The best values are observed when b equals 0.25.

5.4.2 The k_1 Parameter Value

The k_1 parameter has an effect on the term frequency in the document. To determine the best value for k_1, we start by fixing the default parameters (b=0.75, k_3=7) and changing the value of this parameter using possible values that can be used with this parameter according to Robertson and Walker [1999]. We used values between 0 and 7 with an interval of 0.2, and compared results with those obtained using the default value (1.2). The best performance in MAP and R-Precision is seen using values 0.6, 0.4, 0.8, and 1.0. However, this improvement is not significant, and the new values caused a similar reduction in P@10 and Recall. Results are shown in the upper part of Table 5.7.

To check whether this parameter increases performance when using the best value that we previously determined for the parameter b, we investigated values of k_1 with the parameter b set to 0.25. We have also increased the range of values from 7 to 12. Figure 5.5 shows changes in the performance using these values. Overall, all values decrease performance except the

k_3	b	k_1	MAP	P@10	R-Precision	Recall
7	0.75	1.2	0.1803	0.2622	0.2059	**0.6100**
7	0.75	0.4	0.1825	0.2533	**0.2120**	0.5953
		0.6	**0.1830**	0.2556	0.2076	0.6065
		0.8	0.1818	0.2578	0.2091	0.6090
		1.0	0.1810	**0.2656**	0.2072	0.6085
7	0.25	1.2	0.2051	**0.2978**	0.2341	**0.6499**
7	0.25	0.8	0.2048	0.2944	0.2382	0.6397
		1.0	**0.2053**	**0.2978**	**0.2378**	0.6456
		1.4	0.2047	**0.2978**	0.2347	0.6523

Table 5.7: Best results of changing the parameter k_1 in the Okapi BM25 equation. The upper part of the table shows the best results of changing the k_1 value using the default Okapi BM25 b value of 0.75, while the lower part shows the best results obtained by changing the k_1 values while setting the value of the parameter b to 0.25.

value 1 which results in a slight increase. The lower part of Table 5.7 shows the best results recorded using these combinations.

5.4.3 The k_3 Parameter Value

The k_3 parameter controls the impact of term instances appearing in the query. Since we use the titles in our runs, this parameter has no much effects on retrieval. We tried values between 1 and 10 with an interval of 1, and also tried the range of values between 10 and 1000 with an interval of 10, but no improvements are seen either when using the default parameters or using the new b and k_1 values suggested by our earlier experiments.

5.4.4 Parameters with No Stemming

To confirm these results, and to avoid bias to any stemmer or stemming technique, we repeated the same experiments without stemming the collections or the queries. Table 5.8 shows the best values that we obtained. Best results using the default parameters with different b values are shown in the top half of the table, while the bottom part shows the results of tuning the k_1 parameter using the best b value that we determine from the top part (0.2). Results confirm that the best value for the parameter b is at 0.2 and not 0.75.

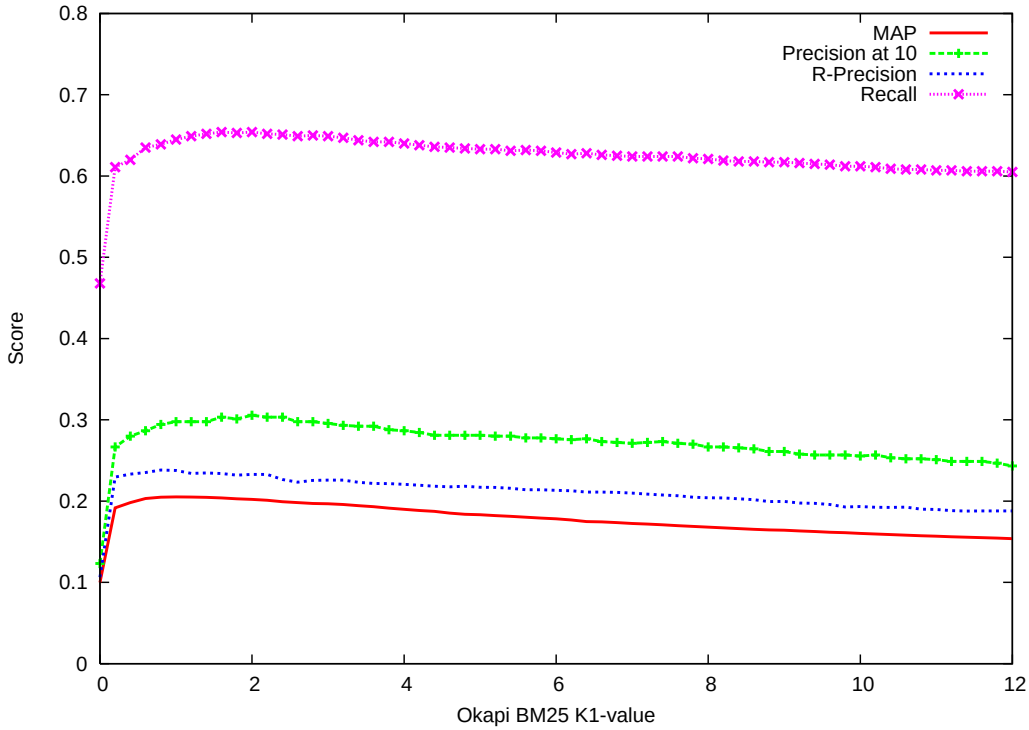

Figure 5.5: The effects of changing the k_1 value using the best b value and the default k_3. Values range from 0 to 12 with an interval of 0.02.

For the unstemmed collection, best performance is obtained for k_1=1.2, but performance is still good at k_1=1.0.

5.5 Tuning TREC 2001 and TREC 2002 Okapi Parameters

We have also explored tuning these parameters for the TREC 2001 and TREC 2002 Arabic collection. In similar experiments, we found that the best value for the parameter b is 0.4, and that the best MAP value is seen when k_1=1.6, and the best P@10 value is seen when k_1=4.4. Table 5.9 shows the best values obtained for the TREC 2001 and TREC 2002 collections. The difference in values between the TREC and AGW collections is due to different collection sizes and different average document lengths. The average document length returned by the Lemur parser when indexing the TREC 2001 and AGW collections without stemming are 168 and 300 terms respectively.

K_3	k_1	b	MAP	P@10	R-Precision	Recall
7	1.2	0.75	0.1414	0.2267	0.1676	0.5572
7	1.2	0.1	0.1724⇑	0.2544⇑	0.2044⇑	0.5961⇑
		0.15	0.1743⇑	0.2656⇑	0.2069⇑	**0.5986**⇑
		0.2	**0.1752**⇑	**0.2700**⇑	0.2061⇑	0.5986⇑
		0.25	0.1741⇑	0.2689⇑	**0.2088**⇑	0.5973⇑
7	1.2	0.2	**0.1752**	0.2700	0.2061	0.5986
7	1.0	0.2	0.1748	**0.2711**	0.2061	0.5934
	1.4		0.1750	0.2667	0.2052	0.6005
	1.6		0.1737	0.2633	**0.2074**	**0.6008**

Table 5.8: Best results of changing the Okapi BM25 equation parameters using the unstemmed collection. The upper part of the table shows the best results of changing the b value using the default Okapi BM25 values, while the lower part shows the best results obtained by changing the k_1 values while setting the value of the parameter b to 0.2. ⇑ indicates values that are significantly better than the default value (0.75) at the 99% confidence level

5.6 Chapter Summary

In this chapter, two main objectives have been achieved. We have built a new test collection of 90 topics with their respective relevance judgements using the AGW document collection. We have used 20 assessors to propose topics and then mark relevant documents in the collection using the ISJ method. Assessors have successfully annotated 122 topics of which we selected 90 topics with their associated relevance judgements.

We used the new test collection to evaluate existing AIR approaches and our proposed techniques. Our results are consistent with those obtained using the TREC 2001 and TREC 2002 topics, with the B.Stem, Al-StemN and light10 stemmers performing the best, and the Khoja heavy stemmer performing the poorest. However, the B.Stem, and Al-StemN perform slightly better than light10 stemmer, but not significantly. When using relevance feedback, the B.Stem and light10 stemmers produce the highest MAP, while Al-StemN and B.Lemma produce the highest recall. Our stemmers show better performance than other stemmers; with relevance feedback, they show lower performance than the light10 and B.Stem algorithms. The difference in MAP, P@10, and R-Precision is not statistically significant.

We have also shown that our corpus-based-lexicon stemmers conflate terms in the corpus

		TREC 2001			**TREC 2002**			**TREC 2001 and 2002**		
b	k_1	MAP	P@10	Recall	MAP	P@10	Recall	MAP	P@10	Recall
0.75	1.2	0.390	0.564	0.670	0.296	0.384	0.760	0.327	0.444	0.723
0.4	1.6	**0.400**	0.624	**0.675**	**0.308**	0.410	0.774	**0.338**⇑	0.481⇑	**0.734**↑
	2.0	**0.400**	0.632	0.673	0.307	0.410	**0.776**	**0.338**⇑	0.484⇑	**0.734**↑
	4.4	0.371	**0.684**	0.641	0.296	**0.426**	0.768	0.321	**0.512**⇑	0.716

Table 5.9: Best Okapi BM25 parameters we determined for the TREC 2001 and TREC 2002 collections. k_3 is set to the default value 7. ↑ indicates values that are significantly better than the default value at the 95% confidence level, ⇑ indicates those significantly better at the 99% confidence level, while ↓ indicates values the are significantly worse than the default value.

more effectively than other algorithms, and that using the corpus as a background lexicon improves both the effectiveness and the efficiency of stemmers that use a professionally prepared lexicon.

The second achievement we have achieved is identifying the best Okapi BM25 parameters for the AGW collection. We have shown that the best value for the b parameter is 0.25, that the best value for the k_1 parameter is 1, and that changing k_3 has no effects on retrieval performance. With the new parameters, performance increased significantly over the default values determined for English documents from the TREC 8 corpus. We have also determined the parameter values that work best for the TREC 2001 and TREC 2002 collections. Our results show that using the default values used for English collections is not the best choice and that the b parameter value affects retrieval effectiveness more than other parameters when using short queries.

We use these parameters in our retrieval experiments in Chapters 6 and 7.

Chapter 6

Foreign Word Identification

The increasing flow of information between languages has led to a rise in the frequency of non-native or loan words, where terms of one language appear transliterated in another. Dealing with such out-of-vocabulary (OOV) words is essential for successful information retrieval. For example, techniques such as stemming should not be applied indiscriminately to all words in a collection, and so before any stemming, foreign words need to be identified. However, in Arabic, foreign words do not follow any consistent format and are written inconsistently, with many versions for the same word appearing in the same document collection [Abduljaleel and Larkey, 2003]. These versions need to be indexed under one term in order to capture documents related to the same term. We must apply special processing on such foreign, non-native Arabic words.

Most of the foreign words that appear in Arabic text are proper nouns. Proper nouns are reported to constitute between 39% and 68% of news queries [Thompson and Dozier, 1997]. Stemming is not a viable means for conflating proper names [Paik et al., 1993]. In some languages such as Indonesian, stemming such words causes around 13% of stemming errors [Asian, 2007]. Orengo and Huyck [2001] stated that stemming proper names in Portuguese is not advisable as recognising proper nouns in Portuguese is not easy due to ambiguity between names and other words. Pfeifer et al. [1996] have also stated that stemming is useful when applied to normal words, but not when applied to proper name searching. We identify foreign words in the text in order to avoid stemming them, and to apply techniques to identify similar variants of the same word. In this chapter, we describe methods of identifying foreign terms in Arabic text.

6.1 Foreign Word Variants

Words borrowed from other languages usually have a different style in writing and construction, and Arabic linguists have drawn up rules to identify them. For example, any root Arabic word that has four or more characters should have one or more of the liquid letters "احرُف الذَلاقة" (ف /f/, ر /r/, م /m/, ن /n/, ل /l/, ب /b/). Those that have no such letters are considered foreign [Al-Shanti, 1996]. However, while such rules could be useful for linguistic purposes, they have limited application in Information Retrieval (IR); based on these rules, many foreign words that have long been absorbed into the language and are spelled consistently would be considered to be foreign. From the IR perspective, we classify foreign words into two general categories: translated and transliterated.

Translated: These are foreign words that are modified or remodelled to conform to Arabic word paradigms; they are well assimilated into Arabic, and are sometimes referred to as Arabicised words [Aljlayl and Frieder, 2002]. This process includes changes in the structure of the borrowed word, including segmental and vowel changes, and the addition, deletion, and modification of stress patterns [Al-Qinal, 2002]. This category of foreign words usually has a single spelling version that is used consistently. Examples include words such as بُستَان (/busṯaan/⟨garden⟩), بُرج (/burʒ/⟨tower⟩), رَادِيُو (/raaḍjuw/⟨radio⟩), and قُنبُلَة (/qunbulaṯ/⟨bomb⟩).

Transliterated: Words in this category are transliterated into Arabic by replacing phonemes with their nearest Arabic equivalents. Although the Arabic language has a broad sound system that contains most phonemes used in other languages, not all phonemes have Arabic equivalents. In practice, such phonemes may be represented in different ways by different persons, resulting in several spelling versions for the same foreign word. For example, we have observed 28 transliterated versions for the name of the former Serbian leader (Milosevic) in the TREC 2002 Arabic collection; these are shown in Table 6.1.

Transliteration has become more common than translation due to the need for instant access to new foreign terms. It can take considerable time for a new foreign term to be included in reference dictionaries. However, users often need to immediately use a particular term, and cannot wait until a standard form of the word is created; news agencies form an important category of such users. This transliteration process often results in multiple spellings in common usage.

ميلوشفيتش	ميلوسيفيتيش	ميليسوفيتش
ميلوشيفيتش	ميليسيفيتش	مليوسيفيتش
مليشيفيتش	ميلسيفيتش	ميلويسفيتش
ميليشفيتش	ميلوسيوفيتش	ميلسوفيتش
ميليشيفتش	ميلوسيفيتس	ميليوسيفيتش
ميلوزيفيتش	ميلوسوفيتس	مليوسوفيتش
ميلوزفيتش	ميلوشيفيتس	مليوسفيتش
ميلوسيفيتش	ميلوسيفتيش	ميلوسوفيتش
ميلوسيفتش	ميلوسفيتش	ميلوسفيتيش
ميلوسفتيش		

Table 6.1: Different spelling versions for the name Milosevic observed in the TREC 2001 Arabic corpus.

6.2 Identifying Foreign Words

We categorise three general approaches for recognising foreign words in Arabic text:

6.2.1 Arabic Lexicons

OOV words can be easily captured by checking whether they exist in an Arabic lexicon. However, the lexicon is unlikely to include all Arabic words, while at the same time it could contain some foreign words. Moreover, this approach will identify misspelled Arabic words as foreign.

We evaluate three approaches that each uses a different dictionary: the Khoja root lexicon [Khoja and Garside, 1999] approach (KLA), the Buckwalter lexicon [Buckwalter, 2002] approach (BLA), and the Microsoft Office 2003 lexicon [Microsoft Corporation, 2002] approach (OLA).

6.2.2 The Arabic Pattern System

We can recognise whether a word is a native Arabic word or a foreign word by comparing it against different patterns. If, after all possible affixes have been removed, the remaining stem matches an Arabic pattern, the word is likely to be an Arabic word. For example, to check whether the word وَالبَاحِث (/walbaaħiθ/⟨and the researcher⟩) is a foreign word, we first remove the prefixes و and الـ to get the stem بَاحِثْ; we find that this word matches

افعوعل	افعلة	افعلال	افعلاء	افعلّ
تفعال	تفاعيل	تستفعل	افعيل	افعولّ
فعالا	فاعول	فاعلة	تفعلّ	تفعلة
فعلّة	فعلة	فعاليل	فعالى	فعالّ
فياعيل	فياعل	فواعيل	فعيلة	فعيلا
مفعلّ	مفعلة	مفعلا	مفعالة	مفاعلة
متفعلّ	فعولة	فعالة	افعول	تفعل
مفعيلا	مفعيل			

Table 6.2: Patterns added to the Khoja modified stemmer to implement the KPA approach.

the pattern فَاعِلْ — it has the same length, and the letter ـا is in the same position — and conclude that it is therefore an Arabic word. Note that we must perform this determination without relying on diacritics.

To use Arabic patterns, we modified the Khoja stemmer to check whether there is a match between a word and a list of patterns after stemming without further checking against the root dictionary. If there is no match, the word is considered a foreign word. We adopted the patterns of the Khoja stemmer and added 37 patterns compiled from Arabic grammar books, these are shown in Table 6.2. We call these approaches the Khoja Pattern Approach (KPA), and Modified Khoja Pattern Approach (MKP) respectively. A word is also considered to be an Arabic word if the remaining stem has three or fewer letters.

We believe that this approach is not perfect, as general Arabic text does not include explicit diacritics; if parts of a foreign word match a pattern, it will be marked as being Arabic. Similarly, misspelled words may be classified as foreign words if no matching pattern is found. Furthermore, pattern matching algorithms are not perfect and falsely extract roots from proper names — including foreign words. This often happens [Larkey et al., 2002].

6.2.3 The n-grams Approach

Transliterated foreign words exhibit construction patterns that are often different from Arabic patterns. By counting the n-grams of a sample of foreign words, a profile can be constructed to identify similar words. This approach has been used in language identification, although it is reported to have only moderate effectiveness in identifying short strings [Cavnar and Trenkle, 1994; Dunning, 1994].

We evaluate the effectiveness of the n-gram method in two ways. First, we extend the n-gram text categorisation method presented by Cavnar and Trenkle [1994]. The method uses language profiles where, for each language, all n-grams that occur in a training corpus are sorted in order of decreasing frequency of occurrence, for n ranging from 1 to 5. To classify a word w, we build its n-gram frequency profile, and compute the distance between each n-gram in the word profile and in each language profile. The total distance is computed by summing up all differences between the position of the n-gram in the word profile and the position of the same n-gram in the language profile. The distance between a word (w) and the Arabic language profile (ALP) is computed as:

$$D_{ALP} = \sum_{i=1}^{N_i} \mid rank(g_i, w) - rank(g_i, ALP) \mid \tag{6.1}$$

where N_i is the number of n-grams in the word w; and $rank$ is the position of gram g_i in the frequency-sorted list of all n-grams for either the word or language profile.

Similarly, the distance between the word (w) and the foreign language profile (FLP) is computed as:

$$D_{FLP} = \sum_{i=1}^{N_i} \mid rank(g_i, w) - rank(g_i, FLP) \mid \tag{6.2}$$

In our work, we build two language profiles, one for native Arabic words and another for foreign words. We compare the n-grams in each word in our list against these two profiles. If the total distance between the word and the foreign language profile is smaller than the total distance between the word and the Arabic language profile, then it is classified as a foreign word. Formally, we determine if a word is foreign if:

$$D_{ALP} - D_{FLP} > 0 \tag{6.3}$$

As the two language profiles are not of the same size, we compute the relative position of each n-gram by dividing its position in the list by the number of the n-grams in the language profile. Figure 6.1 shows the classification process based on this approach. We call this approach the n-gram approach (NGR).

We also try a simpler approach based on the construction of two trigram models: one from Arabic words, and another from foreign words. The probability that a string is a foreign word is determined by comparing the frequency of its trigrams with each language model. A word is considered foreign if the sum of the relative frequency of its trigrams in the foreign words profile is higher than the sum of the relative frequency of its trigrams in the Arabic words profile. We call this approach the trigram approach (TRG).

ALP				WP				FLP		
fr		P_A	$\lvert P_w - P_A \rvert$	fr		P_w	$\lvert P_w - P_F \rvert$	fr		P_F
20300	w	0	0-1	1	b	0	0-2	40025	w	0
20000	b	1	1-0	1	w	1	1-0	39000	f	1
19000	l	2	2-2	1	l	2	2-40	37251	b	2
...	...		3-50	1	bw	3	3-1000		...	
9000	wl	23	4-23	1	wl	4	4-1300	25315	l	40
...	...		5-1000	1	wbl	5	5-41	20012	bwl	41
7000	bw	50						18122	tawl	42
...	...		$D_{ALP} = 1063$				$D_{ALF} = 2370$		...	
1000	bwl	1000						5252	bw	1000
...	...								...	
								1023	wl	1300

Figure 6.1: Using n-grams to identify foreign words. The word "بول" (/bwl/⟨Paul⟩) is categorised as Arabic as $D_{ALP} - D_{FLP} < 0$. ALP is the Arabic language profile, FLP is the foreign words profile, and the WP is the words profile. Profiles are built using the decreasing order of frequency of all grams of size 1 to 5. P_A refers to the position of grams in the Arabic words profile, P_F refers to the position of grams in the foreign words profile, and P_W refers to the position of grams in the word.

6.3 Training Experiments

In this section, we describe how we formed a development data set using Arabic text from the Web, and how we evaluated and improved techniques for identification of foreign words.

6.3.1 Data

To form our development data set, we crawled the Arabic web sites of the Al-Jazeera news channel,[1] the Al-Anwar[2] and El-Akhbar[3] newspapers. A list of 285 482 Arabic words was extracted. After removing Arabic stopwords such as pronouns and prepositions, the list had 246 281 Arabic words with 25 492 unique words.

[1] http://www.aljazeera.net
[2] http://www.alanwar.com
[3] http://www.elkhabar.com

In the absence of diacritics, we decided to remove words with three or fewer characters, as these words could be interpreted as being either Arabic or foreign in different situations. For example, the word بِي /bij/ could be interpreted as the Arabic word meaning ⟨in me or by me⟩, or the English letter ⟨B⟩. After this step, 24 218 unique words remained.

We examined these words and categorised each of them as either an Arabic word (AW), or a transliterated foreign word (FW). We also had to classify some terms as misspelled Arabic words (MW). We used the Microsoft Office 2003 lexicon as a first-pass filter to identify misspelled words, and then manually inspected each word to identify any that were actually correct; the lexicon does not contain some Arabic words, especially those with some complex affixes. The list also had some local Arabic dialect spellings that we chose to classify as misspelled.

The final list had three categories: 22 295 correct Arabic words, 1 218 foreign words and 705 misspelled words.

To build language models for the n-gram approaches (NGR and TRG), we used the TREC 2001 Arabic collection [Gey and Oard, 2001]. We manually selected 3 046 foreign words out of the OOV words extracted from the collection using the Microsoft Office 2003 lexicon. We built the Arabic language model using 100 000 words extracted from the TREC 2001 collection using the same lexicon. We listed all unique words in the collection, and excluded any OOV words, including valid words that do not exist in the lexicon after adding the suffix "ـه" *haa* to them. Unlike most Arabic words, transliterated proper nouns do not appear with this suffix, and so this step guarantees that transliterated proper nouns — even those appearing in the lexicon — will be removed. For example, the proper noun "كريستوفر" (/krjst̪wfr/⟨Christopher⟩) exists in the lexicon, but "كريستوفره" does not, while "كتاب" (/kt̪ab/⟨a book⟩) exists in the lexicon, as does "كتابه" (/kt̪abhu/⟨his book⟩).

6.3.2 Measures of Evaluation

We measure the accuracy of each approach by examining the number of foreign words correctly identified, and the number of incorrect classifications. Based on these numbers, we calculate the precision and recall of each approach. To avoid situations where approaches show better recall than others but have lower precision or vice versa, we use the F_1-measure described in Section 2.3.3 to present the overall performance of each approach. We have also included the MW count to illustrate the effects of misspelled words on each approach

	AW	MW	FW			
Approach	#	#	#	R	P	F
OLA	614	698	1 017	**0.834**	0.437	**0.573**
BLA	384	404	628	0.515	0.443	0.477
KLA	1 732	215	745	0.612	0.277	0.381
KPA	1 034	135	590	0.480	0.340	0.396
MKP	940	126	573	0.470	0.350	0.401
NGR	718	95	726	0.596	**0.471**	0.527
TRG	1 591	118	737	0.605	0.301	0.402

Table 6.3: Initial results of foreign word identification using the Microsoft Office 2003 lexicon (OLA), Buckwalter lexicon (BLA), Khoja root lexicon (KLA), Khoja patterns (KPA), modified Khoja patterns (MKP), n-grams (NGR), and trigrams (TRG). All approaches produce poor precision, with BLA achieving the best precision. OLA has the best recall and is the best performer overall. The # columns indicate the number of items in this category; R is recall; P is precision; and F is the F_1-measure.

6.3.3 Initial Results

Table 6.3 shows results of exposing all words in our list to the different algorithms described in the previous section. We capture all words identified as foreign using each algorithm and then judge them against the actual lists and compute precision, recall and the F_1-measure. The results show that the n-gram approach (NGR) has the highest precision, while the lexicon-based (OLA) approach gives the highest recall. The KPA and MKP pattern-based approaches perform perform well compared to the combination of patterns and the root lexicon (KLA), although the latter produces higher recall. There is a slight improvement in precision when adding more patterns, but recall is sightly reduced. The KLA approach produces the poorest precision, but has a better recall rate than the NGR approach.

The results show that many Arabic native words are identified as foreign words. This is due to two factors: first, a large number of Arabic words is not found in the lexicons we used in the evaluation. This includes Arabic proper nouns and regular Arabic words with complex affixes. Second, n-grams seem to capture a large number of Arabic words due to the lack of diacritics. Some Arabic words are similar in spelling to foreign words but different in pronunciation. Only diacritics would solve the problem of identifying them properly. Our

intention is to conflate different versions of foreign words. Therefore, we try to avoid Arabic words even if they are included in the OOV category as they have, in most cases, unique versions in Arabic text.

Retrieval precision will be negatively affected by incorrect classification of native and foreign words. Consequently, we consider that keeping the proportion of false positives — correct Arabic words identified as foreign (precision) — low to be more important than correctly identifying a higher number of foreign words (recall).

Some of the Arabic words categorised as foreign are in fact misspelled; we believe that these have little effect on retrieval precision, and there is limited value in identifying such words in a query. These may be better handled by a spelling correction stage in the retrieval system.

6.4 Improving Results

With the current results, none of the above approaches are suitable for identifying foreign words, and therefore, improvement is essential. We used Arabic grammar rules, Arabic letters and words frequency, n-gram profile size, and a combination of these approaches to improve results. In this section we present improvements to these approaches.

6.4.1 Enhanced Rules

To reduce the false identification rate for foreign words, we analysed the lists of foreign words, correct Arabic words identified as foreign, and misspelled words identified as foreign. We noticed that some Arabic characters rarely exist in transliterated foreign words, and used these to distinguish Arabic words — correctly or incorrectly spelled — from true foreign words. Table 6.4 shows the count of each character in the sample of 3 046 foreign words; foreign words tend to have vowels inserted between consonants to maintain the CVCV[4] paradigm. We also noticed that most of transliterated foreign words do not start with the definite article "ال", or end with the Taa Marbuta "ة". Foreign words also rarely end with two Arabic suffixes.

We also noticed that lexicon-based approaches fail to recognise some correct Arabic words for the following reasons:

- Words with the letter *alef* ("ا") with or without the diacritics *hamza* ("أ", "إ"), or the

[4] "C" stands for a consonant, and "V" stands for a vowel.

Letter	count	letter	count	letter	count
ي	3 839	م	632	ح	2
ا	3 599	د	559	ع	2
و	2 453	ش	514	ص	1
ن	1 660	ج	458	ء	0
س	1 587	ز	334	وء	0
ت	1 544	ه	171	أ	0
ر	1 244	خ	84	إ	0
ك	1 070	ث	23	آ	0
ب	900	ق	20	ض	0
ل	863	ط	12	ظ	0
ف	769	يء	7	ى	0
غ	728	ذ	3	ة	0

Table 6.4: Frequency of Arabic letters in a sample of 3 046 foreign words.

diacritic *madda* ("آ") are not recognised as correct in many cases. Many words are also categorised incorrectly if the *hamza* is wrongly placed above or below the initial *alef* or if the *madda* is absent. In modern Arabic text, the *alef* often appears without the *hamza* diacritic, and the *madda* is sometimes dropped.

- Correct Arabic words are not recognised with particular suffixes. For example, words that have the object suffix, such as the suffix "ها" in يُعَلِمونَكَهَا (/juʕallimwnakahaa/⟨they teach it to you⟩).

- As described in Section 4.2.2, some Arabic words are compound words, written attached to each other most of the time. For example, compound nouns composed of two words that are individually identified as being correct, such as عَبدُالقَادِر (/ʕabd̪ualqaad̪ir/⟨Abdulqader⟩), are flagged as incorrect when combined.

- Some common typographical shortcuts result in words being written without whitespace between them. Where a character that always terminates a word (for example "ة") is found in the apparent middle of a word, it is clear that this problem has occurred.

From these observations, we constructed the following rules. Whenever one of the follow-

	AW	MW	FW			
Approach	#	#	#	R	P	F
OLA	145	248	866	**0.711**	0.687	**0.699**
BLA	88	149	534	0.438	**0.693**	0.537
KLA	420	83	642	0.527	0.508	0.543
KPA	302	52	520	0.427	0.595	0.497
MKP	269	51	507	0.416	0.613	0.496
NGR	411	69	669	0.549	0.582	0.565
TRG	928	85	642	0.527	0.387	0.447

Table 6.5: Improvements added using our rules: identification is increased on all approaches. The OLA approach outperforms all other approaches. The # columns indicate the number of items in this category; R is recall; P is precision; and F is the F_1-measure.

ing conditions is met, a word is not classified as foreign:

1. the word contains any of the Arabic characters:
   يء, ء, ذ, ح, ص, أو, أ, إ, آ, ظ, ض, ى, or ة;
2. the word starts with the definite article (الـ);
3. the word has more than one Arabic suffix (pronouns attached at the end of the word);
4. the word has no vowels between the second and penultimate characters (inclusive); or
5. the word contains one of the strings:
   ة, ى, ء, اا, يال, رال, زال, دال, ذال, وال, or االا;
   and when split into two parts at the first character of any sequence, the first part contains three or more characters, and the second part contains four or more characters.

Table 6.5 shows the improvement achieved using these rules. It can be seen that they have a large positive impact. Overall, OLA was the best approach with precision at 69% and recall at 71%. Figure 6.2 shows the precision obtained before and after applying these rules. Improvement is consistent across all approaches, with an increase in precision between 10% and 25%.

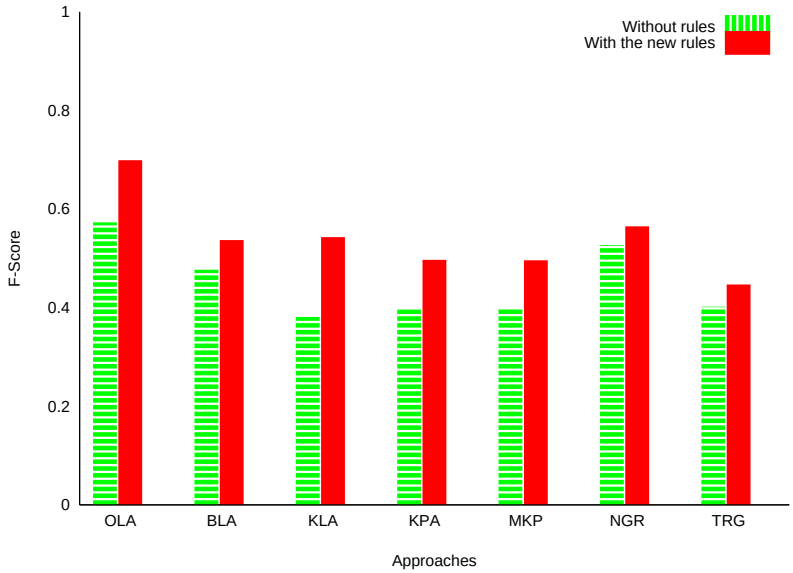

Figure 6.2: Precision of different approaches with and without our new rules. Improvements are consistent across all approaches

6.4.2 Improving the n-gram Approach

In the preceding section we used the n-gram approach without checking the best profile length for Arabic, nor did we test different word profile sizes. To avoid confusion, we use the term "profile size" to represent the size of grams used to build the language profile, and we use the term "profile depth" to represent the total number of grams included in the language profile, usually the most n frequent grams. For example, a profile size of 4-grams includes all grams from 1 to 4 ordered by decreasing frequency, and a profile depth of 500 consists of the first 500 grams of that profile. In the previous section, we used the complete language profile for both foreign words and Arabic words and computed the distance by subtracting the position of the gram in the word from the relative position — the gram position divided by the profile length — of the same gram in the language profiles. This differs from the approach of Cavnar and Trenle, who used the top 300 ranked n-grams of each profile. They stated that around that rank, n-grams are more specific to the subject of the document and represent terms that occur very frequently in the document around the subject, (in our case foreign words). By inspecting the language profile, they concluded that a better cutoff can be chosen.

		AW	MW	FW			
Word Profile Size	Language Profile Depth	#	#	#	R	P	F
5-grams	All	718	95	726	0.596	**0.471**	0.527
2-grams	2 500	1 243	139	873	0.717	0.387	0.503
3-grams	1 700	1 315	156	1 017	**0.835**	0.409	**0.548**
4-grams	1 200	1 449	157	1 000	0.821	0.384	0.523
5-grams	900	1 546	158	1 002	0.823	0.370	0.511

Table 6.6: Best word profile size and the language profile depth at which the best results are recorded. The # columns indicate the number of items in this category; R is recall; P is precision; and F is the F_1-measure.

In this section we aim to determine the most appropriate language profile size and depth that can be used to identify foreign words. We also determine the cutoff value that leads to the best result in identifying foreign words. For each word in the list we generate grams from 1 to n where n ranges from 1 to 6, and rank them by frequency. We compute distance as before. To decide on the best depth that can be used to generate word profiles, we run the algorithm with different depths starting at the most frequent gram and stopping at the mth gram in the language profile. We run experiments with m ranging from 100 to 16000. Figure 6.3 shows the F_1-measure recorded across the language profile depths using the development data set. Table 6.6 shows the best results achieved by the different language profile depths, compared to using the full language profile as a baseline. The optimal cutoff value for determining foreign words appears to depend on the number of grams used to build the word profile. Results show that while the profile size increases, the profile depth that produces the best result decreases. The best results produced by different profile sizes were similar, with grams of size 1 to 3 achieving the best results. With these results, in order to achieve efficiency, a profile size of 1 to 5-grams is the best option. However, as our objective is effectiveness, we choose to build the word profile using grams of size 1 to 3 and limit the language profiles depth to the first 1 700 most frequent grams. This option outperforms the initial result obtained using the whole language profiles, and is used as the baseline of our next experiment to improve the cutoff value at which we determine that a word is foreign.

In the previous section, we decided that a word is foreign if its distance from the foreign language profile is shorter than its distance from the Arabic profile. For instance, if an Arabic word has a distance of 300 to the Arabic profile and a distance of 299 to the foreign profile, it

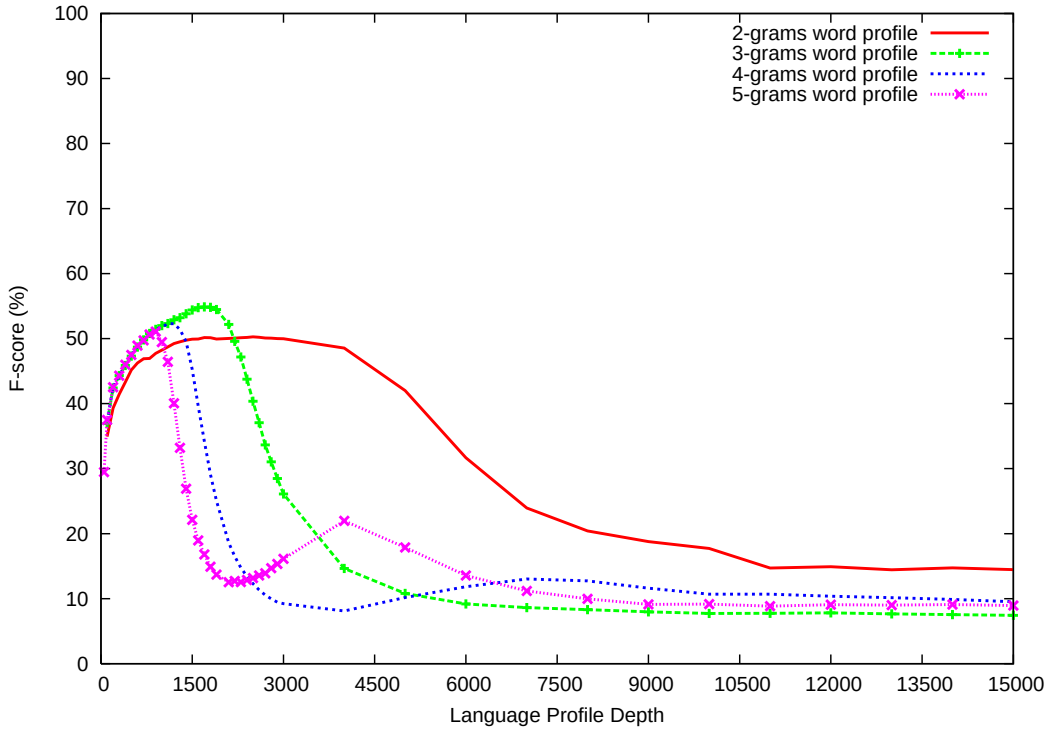

Figure 6.3: The effects of number of grams used in the word profile and the depth of language profile on foreign word identification. Word profile built using grams from 1 to 3 gives the best results when the language profile depth is 1 700.

is classified as foreign. To avoid such borderline cases and to increase precision by minimising the number of Arabic words being identified as foreign, we increase the threshold required for a word to be considered foreign. The optimal cutoff value needs to be determined. With equal-sized language profiles, we calculate the distance between a word w and the Arabic profile and the distance of the same word and the foreign profile as shown in Equations 6.1 and 6.2 respectively, and classify a word as foreign only when:

$$D_{ALP} - D_{FLP} > c \tag{6.4}$$

where c is the cutoff value between the two profiles. Using language profiles of depth 1 700, and building word profiles with grams of size 1 to 3, we calculate the distance between words in our list and both language profiles using different cutoff values. We determine that the best cutoff value for this data set is 2 000. Table 6.7 shows the number of Arabic, misspelled, and foreign words identified using this threshold. Choosing the right profile size, depth, and

	AW	MW	FW			
	#	#	#	R	P	F
NGR	718	95	726	0.596	0.471	0.527
1700LP c=0	1 315	156	1 017	**0.835**	0.409	0.549
1700LP c=2000	437	84	810	0.665	**0.609**	**0.636**

Table 6.7: Improvements in precision by choosing the best cutoff value. NGR is the initial n-gram approach using the complete language profiles where n ranges from 1 to 5, 1700LP stands for using a profile of depth 1 700 with a profile size 3, and c is the cutoff value. The # columns indicate the number of items in this category; R is recall; P is precision; and F is the F_1-measure. 1700LP c=0 is the optimal approach from Table 6.6.

	AW	MW	FW			
	#	#	#	R	P	F
1700LP c=0	2198	170	1120	**0.921**	0.321	0.476
1700LP c=2000	556	65	803	0.659	**0.564**	**0.608**

Table 6.8: Effects of stemming on the n-gram approach. Stemming increases recall of the n-gram approach at cutoff 0, but decreases precision. The # columns indicate the number of items in this category; R is recall; P is precision; and F is the F_1-measure.

cutoff value increased precision over the initial n-gram approach.

Figure 6.4 shows the difference between the distance of words to the Arabic language profile and their distance to the foreign language profile. The figure shows that most foreign words are above the 0 line (c=0). The best precision is observed for c=2 000. Figure 6.5 shows the effect of changing the cutoff value on results of the first data set.

Improving the n-gram Approach Using Stemming

Native Arabic words exhibit a different gram profile from stemmed Arabic words. The most frequent grams in the language profile usually contain the language letters, and the most frequent affixes of the language [Cavnar and Trenkle, 1994]. With stemming, we remove affixes from words, thus removing the top-ranked grams in the language profile. This would result in building a language profile based on the language roots or stems. To check the effects of stemming on the n-grams identification technique, we stemmed the collections and

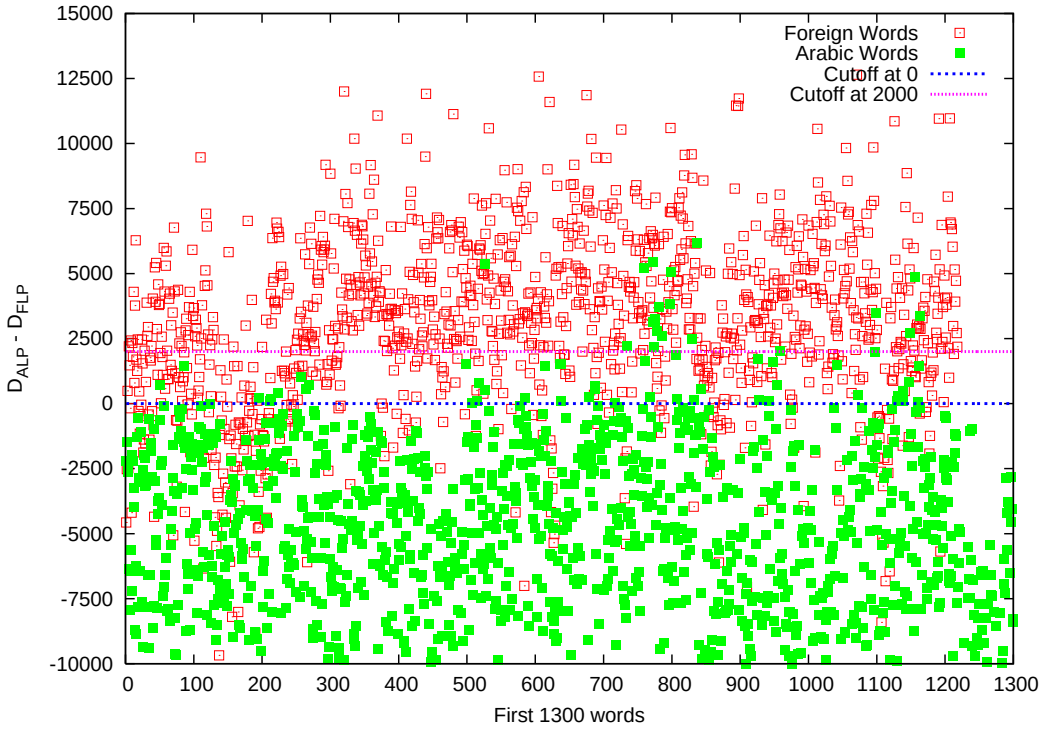

Figure 6.4: The difference between the distance from a word profile to the Arabic language profile (D_{ALP}) and the distance from the same word to the foreign language profile (D_{FLP}). The cutoff that captures the most foreign words is 0, and the cutoff that gives the best precision is at 2000.

built language profiles using the stemmed collections. We also stemmed the three lists that we classified in our data set, and generated the unique list from these. Table 6.8 shows results of using the n-gram approach on the stemmed collection. The precision of the n-gram approach decreases when stemming the collection, but recall increases.

6.5 Using Word Frequency and Stemming to Identify Foreign Words

Word frequency can be used as an indicator to determine foreign words in Arabic text. Foreign words generally appear less frequently than native words in Arabic text, although naturally there are some very common foreign words, and some very rarely-used native words, particularly in the context of news. We believe that word frequency can be used to filter out very frequent words before we examine whether a word is foreign.

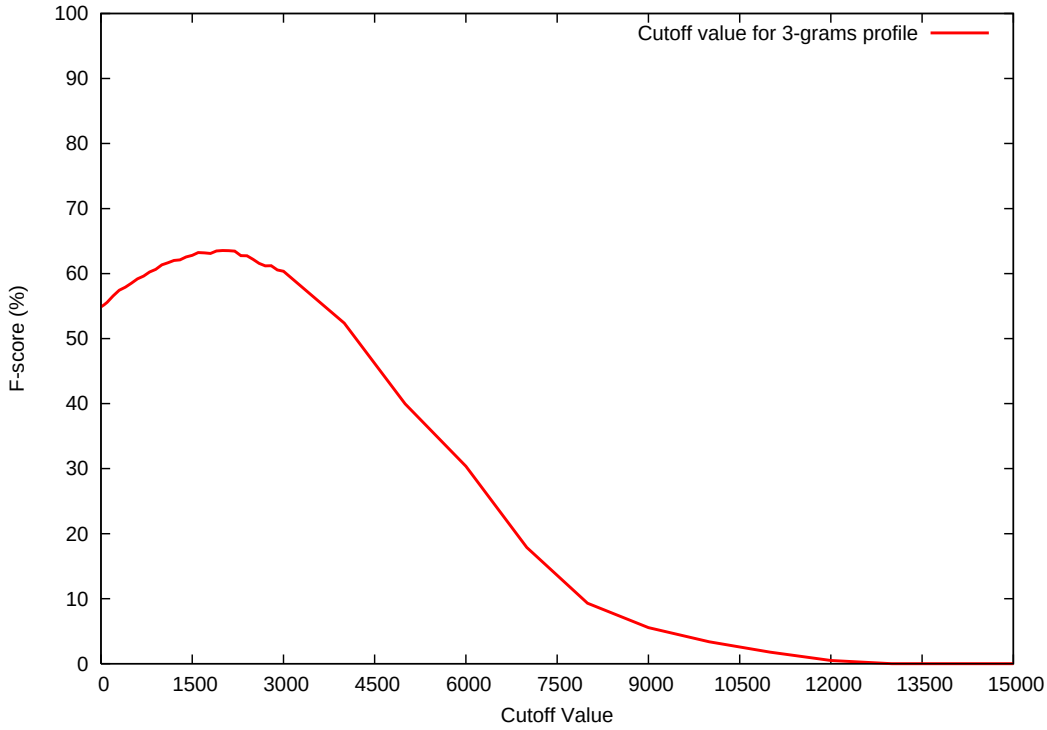

Figure 6.5: Effects of cutoff values on identifying foreign words. The best F-score value is seen at cutoff 2000, when building words profile using grams from 1 to 3 and using the most 1 700 frequent grams in language profile.

To determine the effects of using word frequency in identifying foreign words, we count occurrences of Arabic, foreign, and misspelled words in their original crawled collection (described in 6.3.1) using a frequency threshold from 1 to 600. The left side of Table 6.9 shows the numbers of words in our data set that occur at different frequencies; there is a large overlap in the frequency of both Arabic and foreign words. As we expected, 75% (912) of foreign words occur fewer than four times in our data set. However, the number of Arabic native words below this threshold is also high (15 254). Considering the threshold where all foreign words can be captured — that is, which words occur fewer than 500 times — the number of Arabic words would increase to 22 266. As Arabic words are highly inflected, and foreign words are usually nouns that do not accept most Arabic affixes, stemming should increase the frequency of Arabic words, and consequently enable the identification of foreign words at lower frequency levels. We stemmed the whole data set (Arabic words, foreign words and

Frequency Threshold	Occurrences AW	FW	MW
1	8257	579	339
2	13 277	832	547
3	15 254	912	595
4	16 650	964	631
5	17 639	1 003	642
6	18 303	1 033	653
7	18 749	1 045	656
8	19 117	1 058	662
9	19 391	1 062	665
10	19 653	1 066	668
20	20 809	1 099	678
30	21 247	1 172	684
40	21 520	1 192	687
50	21 653	1 196	688
100	21 883	1 206	689
200	22 168	1 212	705
300	22 206	1 216	706
400	22 245	1 217	706
500	22 266	1 218	706
600	22 273	1 218	706

Frequency Threshold	Occurrences AW	FW	MW
1	3 844	488	261
2	6 257	719	425
3	7 347	804	487
4	8 105	849	527
5	8 649	886	539
6	9 031	923	554
7	9 335	938	563
8	9 603	954	570
9	9 756	959	575
10	9 936	965	582
20	10 833	1 009	602
30	11 212	1 075	612
40	11 466	1 107	620
50	11 611	1 115	623
100	11 910	1 127	629
200	12 195	1 134	662
300	12 270	1 138	667
400	12 311	1 139	671
500	12 331	1 140	672
600	12 348	1 140	673

Table 6.9: Arabic and foreign word frequencies: Occurrences before stemming (left) and after stemming (right). Stemming affects 44.40% of Arabic words, while affecting only 6.40% of foreign words.

misspelled words), generated the unique list after stemming, and computed word frequency again. This process left 123 96 Arabic words, 1 140 foreign words, and 675 misspelled words. The right side of Table 6.9 shows word frequencies after stemming. While stemming slightly increases frequency statistics for Arabic words, and does not affect the corresponding statistics for foreign words, we find that for this data set, word frequency alone does not help to distinguish foreign words.

To confirm these results, we tested our scheme on a bigger collection. We counted the frequency of our word lists in the TREC 2001 Arabic collection. We first extracted Arabic, foreign, and misspelled words that exist in the TREC 2001 collection from our three lists.

Frequency Threshold	Occurrences AW	FW	MW
1000	16 235	758	390
2000	18 000	829	399
3000	18 852	863	402
4000	19 304	889	405
5000	19 569	899	405
6000	19 758	906	405
7000	19 913	911	406
8000	20 026	913	406
9000	20 118	916	406
10000	20 193	918	406
11000	20 257	919	406
12000	20 315	920	406
13000	20 346	921	406
14000	20 379	921	406
15000	20 407	921	406

Frequency Threshold	Occurrences AW	FW	MW
1000	6 789	669	414
2000	8 088	763	446
3000	8 756	828	463
4000	9 157	864	471
5000	9 447	884	474
6000	9 659	901	479
7000	9 818	912	481
8000	9 965	916	483
9000	10 070	920	486
10000	10 164	922	488
11000	10 244	925	489
12000	10 309	931	492
13000	10 365	932	493
14000	10 418	935	494
15000	10 460	936	495

Table 6.10: Arabic and foreign word frequencies using TREC 2001 collection: Occurrences before stemming (left) and after stemming (right).

Using our list of 22 295 Arabic words, 1 218 foreign words, and 705 misspelled words; we found 20 730 Arabic words, 930 foreign words, and 406 misspelled words in the TREC 2001 collection. We use these frequencies to help distinguish foreign words. We also stemmed the collections and the new lists and counted the word frequencies after stemming. Table 6.10 shows the word frequency for Arabic, foreign and misspelled words before and after stemming. These results show that word frequency cannot be used by itself to identify foreign words in Arabic. However, they do show that stemming greatly helps in distinguishing Arabic words, and can therefore be used to improve precision when identifying foreign words. Results on the first data set show that stemming reduces the number of Arabic words from 22 295 to 12 396; stemming affects 44.40% of Arabic words, but only 6.40% (78) of foreign words.

6.6 Combining Approaches

In this section, we apply a combination of the above approaches to identify foreign words. We used approaches that produce high recall to minimise Arabic words and pass results to

	AW	MW	FW			
	#	#	#	R	P	F
n-grams0 and OLA	72	156	872	**0.716**	0.793	**0.752**
n-grams2000 OLA plus rules	59	123	804	0.660	0.815	0.729
n-grams2000 and OLA	42	83	713	0.585	**0.851**	0.694
n-grams0 and BLA	43	88	534	0.438	0.803	0.567

Table 6.11: Combining n-grams and lexicon approaches: n-grams0 refers to the n-gram approach with a cutoff value 0, and n-grams2000 refers to the n-gram approach with a cutoff value 2000. The n-grams0 technique combined with the Microsoft Office 2003 lexicon produces the best result.

	AW	MW	FW			
Approach	#	#	#	R	P	F
OLA	1 189	112	417	**0.777**	0.242	0.370
BLA	780	96	267	0.498	0.234	0.318
KLA	1 684	55	312	0.582	0.152	0.241
KPA	992	29	238	0.440	0.189	0.265
MKP	901	26	231	0.431	0.199	**0.273**
NGR	740	22	286	0.533	**0.272**	0.361
TRG	1 655	19	308	0.575	0.155	0.245

Table 6.12: Identification of foreign words on the test set: initial results.

approaches that produce high precision in distinguishing foreign words.

We passed foreign words identified by the n-gram approach with cutoff values 0 and 2000 to the Microsoft Office 2003 lexicon, and Buckwalter lexicons. We also combined the n-gram approach with the OLA approach after using our enhancement rules. Table 6.11 presents results of these combinations. The n-gram approach plus the Microsoft Office 2003 lexicon captures about 71% of foreign words at a precision of 79%. This result is even better than using the Microsoft Office 2003 lexicon with our enhanced rules, or using OLA alone (Table 6.3).

	AW	MW	FW			
Approach	#	#	#	R	P	F
OLA	302	38	307	**0.572**	0.474	**0.519**
BLA	149	33	184	0.343	**0.502**	0.408
KLA	350	16	216	0.403	0.371	0.386
KPA	238	9	166	0.310	0.402	0.350
MKP	202	8	162	0.302	0.435	0.357
NGR	401	8	245	0.457	0.374	0.412
TRG	972	11	235	0.438	0.193	0.268

Table 6.13: Identification of foreign words on the test set: results after using the new rules.

6.7 Verification Experiments

To verify our results, we used two other data sets. We collected a list of 23 466 unique words from the Dar-al-Hayat newspaper.[5] We classified and marked words in the same way as for the first data set (described in Section 6.3.1). We determined this new set to comprise 22 800 Arabic words (AW), 536 Foreign words (FW), and 130 Misspelled words (MW). Table 6.12 and Table 6.13 show the initial results and improvements using the enhanced rules obtained by each approach using this data set. The results on this unseen data are relatively consistent with the previous experiment, but precision in this sample is lower. Using this data set, we confirmed that the best language profile depth at which this approach produces the highest F_1-measure value is 1 700 when using a word profile of size 3, and the best cutoff value at which it produces the best result is 2 000. The best recall value is observed at a cutoff value of zero.

Combining the n-gram approach and the Microsoft Office 2003 lexicon approach produced the best precision and recall values. Table 6.14 shows results of running both the n-gram and OLA on the collection.

To form our third data set, we used 3 925 manually transliterated foreign words. The transliteration process is described in Section 7.1.2. We mixed these words with the Arabic and misspelled words from the second data set and evaluated the approaches on this larger — albeit not completely independent — data set. Table 6.15 shows results of running the n-gram and OLA approaches. Using the n-gram approach with a cut-off 0 and OLA, we

[5] http://www.daralhayat.com

	AW	MW	FW			
	#	#	#	R	P	F
n-grams0 and OLA	99	24	337	**0.629**	0.733	**0.677**
n-grams2000 and OLA	43	4	256	0.478	**0.845**	0.610

Table 6.14: Combining n-grams and lexicon approaches using the second data set: n-grams0 refers to the n-gram approach with a cutoff value 0, and n-grams2000 refers to the n-gram approach with a cutoff value 2000. The n-grams0 technique combined with the Microsoft Office 2003 lexicon produces the best result.

	AW	MW	FW			
	#	#	#	R	P	F
n-grams0	1 298	155	3 534	**0.900**	0.709	0.793
n-grams2000	426	84	2 834	0.722	0.848	0.780
n-grams0 and OLA	70	155	3 169	0.807	0.934	**0.866**
n-grams2000 and OLA	40	84	2 593	0.660	**0.954**	0.781

Table 6.15: Results using combined approaches of n-grams and OLA approach using the third data set: n-grams0 refers to the n-gram approach with a cutoff value 0, and n-grams2000 refers to the n-gram approach with a cutoff value 2000. The n-grams0 technique combined with the Microsoft Office 2003 lexicon produces the best result.

identified 80% of foreign words with a precision of 93%.

6.8 Effects of Foreign Word Identification on Retrieval Performance

To check whether identification of foreign words has an effect on retrieval performance, we extracted all words identified as foreign out of the list of unique words of the AGW collection using both OLA and the n-grams approach with a cutoff valued-zero. To minimise the misspelled words identified as foreign, we used our normalisation and SureSplit techniques described in Section 4.2.2 for both the queries and the collection. We used the identified foreign words list as an "unstemmable" word list with both the light11 algorithm and the Khoja root stemmer. A word that exists in that list is returned without stemming. Words in the queries are also stemmed the same way. Table 6.16 shows results of running both algorithms with and without the unstemmable list of foreign words.

Technique	MAP	P@10	RP	RECALL
light11	0.2053	0.2978	0.2378	0.6456
light11 with FW unstemmed	0.2039	0.2956	0.2371	0.6454
light11 with FW initial prefix removed	**0.2086**	**0.3022**	**0.2399**	**0.6627**
Khoja	0.1654	0.2544	0.1988	0.5773
Khoja With FW unstemmed	0.1645	0.2533	0.1945	0.5502
Khoja with FW initial prefix removed	**0.1707**	**0.2633**	**0.2030**	**0.5939**

Table 6.16: Effects of not stemming foreign words on retrieval performance based on our combined OLA and n-grams0 identification approach. Not stemming foreign words decreases the performance of both root and light stemmers. However, removing the first prefix from foreign words, improved both stemmers but not significantly.

Results show that excluding foreign words from stemming did not improve retrieval. In fact, the performance of both stemmers is affected slightly negatively. As most frequent affixes in foreign words are conjunctions and prepositions, which occur at the beginning of the word, we conducted another experiment where we returned the remaining string after the first letter if it existed in the foreign words list, and returned the whole foreign word otherwise. We did this with both stemmers for the whole collection and the queries. Results show that removing the first letter improves both the light stemming and the root stemming. The improvement is insignificant for both stemmers.

6.9 Discussion

We have seen that foreign words are not easily recognised in Arabic text, and a large number of Arabic words are affected when we try to identify foreign words and exclude them from further morphological operations such as stemming.

We found the lexicon approach to be the best in identifying foreign words. However, current lexicons are relatively small, and the variety of Arabic inflection makes it very difficult to include all correct word forms. Furthermore, current lexicons include many foreign words; for example when using the OLA approach on the first data set, 1 017 foreign words out of 1 218 are OOV, indicating that about 200 foreign words are present in that lexicon. The pattern approach is more efficient, but the lack of diacritics in general written Arabic makes it very difficult to precisely match a pattern with a word; this results in many foreign words

being incorrectly identified as Arabic. When passing the list of all 3 046 manually judged foreign words to the pattern approach, some 2 017 words of this list were correctly judged as foreign, and about one third (1 029) were incorrectly judged to be Arabic. The n-gram method produced reasonable precision compared to the lexicon-based methods. In contrast, TRG had the worst results. This could be due to the limited size of the training corpus. However, we expect that improvements to this approach will remain limited due to the fact that many Arabic and foreign words share the same trigrams.

All the approaches are improved dramatically when applying the enhancement rules. The improvement was less marked for the NGR approach, since it does apply some of the rules such as letter counts implicitly. The lack of diacritics also makes it very difficult to distinguish between certain foreign and Arabic words. For example, without diacritics, the word كلينتين could be كِلِينتُن (/klijnṯun/⟨Clinton⟩), or كَلِينَتَين (/kalijnaṯajn/⟨as two date trees⟩). The pronunciation is different in the two cases, but only context or diacritics can distinguish the word.

By determining the best language profile depth and using a word profile of size 3, we improved the identification using the n-gram ranked approach. By combining the OLA approach with the n-grams approach, we achieved a recall of 80% with a precision of 93% when using a manually transliterated word list embedded within typical Arabic text. This result is even better than results with OLA and our enhancement rules. We relate this improvement to the fact that many Arabic words are filtered out by the n-grams approach before we check them with the OLA approach. This minimises the number of Arabic words that OLA incorrectly distinguishes as foreign.

Identifying foreign words allows us to avoid stemming them along with native Arabic words. Results show that not stemming foreign words results in a slight reduction in precision for the light11 stemmer and the Khoja stemmer. When removing the first letter from foreign words that exist in the list without that letter, results improved, although this improvement is not significant for the light stemmer and the root stemmer.

6.10 Chapter Summary

Identifying foreign words in Arabic text is an important issue in information retrieval, hence commonly-used techniques such as stemming should not be applied indiscriminately to all words in a collection.

We have examined three approaches for identifying foreign words in Arabic text: lexicons,

patterns, and n-grams. We have presented results that show that the lexicon approach outperforms the other approaches, and have described rules to minimise the false identification of foreign words. These rules result in improved precision, but have a small negative impact on recall.

We have shown that the word frequency cannot be used by itself to identify foreign words in Arabic text even after stemming, but that it can be used to reduce the number of Arabic words involved in the checking process. We have explored how to improve the n-gram approach by determining the best language profile depth and size. We have formed the best language profile from the 1 700 most frequent n-grams for grams of size 1 to 6; and have improved the identification effectiveness of the original n-gram approach used in language identification. By increasing the threshold at which we decide a word is foreign from 0 to 2 000, we have improved precision, but at the cost of recall.

We have combined approaches to improve results. We selected approaches that have higher recall values and precision. The n-gram approach, in conjunction with the Microsoft Office 2003 lexicon, OLA, and a cutoff of 0 produces results better than even our rule-based approach.

We combined the OLA and the n-gram approaches to capture the list of foreign words in the AGW collection and used this list as an unstemmable list with both the light11 light stemmer, and the Khoja root stemmer. Any word found in that list is returned without removing affixes. We found that not stemming foreign words does negatively affect the precision of light and root stemmers, suggesting that removing affixes such as conjunctions and prepositions is essential. We further removed the first letter from foreign words if the remaining strings exist in the identified foreign word list. This improves the performance of both light and root stemmers but not significantly.

Since foreign words may have several variants, algorithms that collapse those versions to one form could be useful in identifying foreign words. Given a foreign word in the query, algorithms such as string- and phonetic-similarity techniques could be used to identify variants of the word in the query and either replace them with the version found in the query or normalise them to one form in the collection. We present such techniques in the following chapter and show how identifying foreign words and normalising all variants in the collection can aid retrieval effectiveness.

Chapter 7

Dealing with Foreign Words in Arabic

Due to inconsistent transliteration, foreign words frequently have many variants in Arabic text. As explained in the previous chapter, transliterated foreign words are increasingly common in Arabic text, and there is little published research on how to deal with them. Typical search engine users are unlikely to recognise the problem, and rarely use variants in their queries. Currently, major search engines such as Google, Yahoo, and Microsoft Live Search use exact match for Arabic search, and no publicly available AIR system has been reported to retrieve different spelling variants [Abdelali et al., 2004].

In this chapter, we explore how the different variants of a foreign word may be captured and conflated together. We test existing similarity techniques described in Section 2.2.3, and introduce three techniques to search for variants of foreign words in Arabic. In the first technique, we convert different variants to a single normalised form by removing vowels and conflating homophones. In the second technique, we extend the well-known Soundex technique — commonly used to identify variants of names in English — to the foreign words problem in Arabic. In the third technique, we modify the English Editex algorithm to identify similar foreign words in Arabic. We use these techniques in an IR experiment and show that our novel algorithms improve results.

بيكم	بوكم	باكم	بكم
بيكوم	بوكوم	باكوم	بكوم
بيكام	بوكام	باكام	بكام
بيكيم	بوكيم	باكيم	بكيم

Table 7.1: Variants of the word "Beckham" generated by adding vowels.

7.1 Data

To test the effectiveness of our algorithms, we use two different data sets. The first set is generated from text crawled from the Web, and the second is prepared by manual transliteration of foreign words from English to Arabic.

7.1.1 Crawled Data

This set is derived from a one-gigabyte crawl of Arabic web pages from twelve different online news sites. From this data we extracted 18 873 073 Arabic words, 383 649 of them unique. We used the Microsoft Office 2003 Arabic lexicon to build a reference list of OOV words. To avoid duplicates in the 40 514 OOV words returned by the lexicon, we remove the first character if it is an Arabic preposition and the string remaining after that character exists in the collection. We also removed the definite article "ال" to obtain a list of 32 583 words. Through manual inspection, we identified 2 039 unique foreign words.

To evaluate alternative techniques, we use a reference list of foreign words and their variants. To identify variants, we generate all possible spelling variants of each word according to the patterns we described in Section 2.1.5, and kept only the patterns that exist in our collection; 556 clusters of foreign words remain.

Generation of Variants

To generate foreign words variants, we first remove any vowels and then reinsert vowel combinations of the three long vowels {ا ي و} between the consonants that remain. For a word of length n, this process generates $4^{(n-1)}$ variants. Consider the word بيكام ⟨Beckham⟩. We remove vowels to obtain بكم, and then add all possible vowels to obtain the variants shown in Table 7.1.1.

As discussed in Section 2.1.5, inconsistent representation of sounds between transliterators adds to the variations in spelling. Thus, the number of possible transliterations for a foreign

word is given by $4^{(n-1)}$ multiplied by the number of possible transliterations for each of its consonants. In our example, the letter ـقـ /q/ may also be used in place of ـكـ /k/, and so we generate another set using that letter.

We validate the generated variants against our collection and keep only those that appear in the crawled text. For our example word "Beckham", we found only two correct variants: بيكام and بيكم. Some of the generated variants could be correct Arabic words that would be valid when checked against the collection. Many of the generated clusters were found to be noisy – that is, they included many native Arabic words. We manually corrected these clusters by removing unrelated Arabic words. The average cluster length is 2.8 words; the smallest cluster has two variants, and the largest has nine, with a total of 1 718 words.

7.1.2 Transliterated Data

Our second collection reflects one pattern in which OOV words are introduced by ordinary users transliterating English words into Arabic. We extracted a list of 1 134 foreign words from the TREC 2002 Arabic collection, and passed these to the Google translation engine to obtain their English equivalents. We manually inspected these and corrected any incorrect translations. We also removed the 57 words mapped by Google to multiple English words. These are usually a word and a possible conjunction or preposition. For example the word لوكسمبرج ⟨Luxembourg⟩ is incorrectly translated to ⟨for June⟩. We passed the English list to seven native Arabic speakers and asked them to transliterate each word in the list back into Arabic, even if the word has an Arabic equivalent. At the time of the experiment, four were PhD students and had finished an advanced-level English course, three were enrolled in an intermediate-level English course. Participants were asked to type in their transliteration next to each English word. We noticed that some transliterators had only basic computing skills, and made many spelling mistakes. For example, instead of typing the letter Alef "ا", we found that transliterators sometimes mistakenly type the letter Lam "ـلـ"; this is analogous to users mistakenly interchanging "0" and "O", and "1" and "l" in English.

We clustered transliterations by the original English words, removed duplicates from each cluster, and also removed 103 clusters where all transliterators agreed on the same version of transliteration. This left 3 582 words in 207 clusters of size 2; 252 clusters of size 3; 192 clusters of size 4; 149 clusters of size 5; 93 clusters of size 6; and 47 clusters of size 7. Finally, we incorporated these transliterations into a list with 35 949 unique Arabic native words that we used in the previous chapter (Sections 6.3.1 and 6.7).

7.2 Algorithms

There are two types of algorithms that we can use to find variants of a foreign word that appears in the user query: techniques that can be used at the indexing time — known as *static* techniques — and techniques that can be used at search time, known as *dynamic* techniques. In static techniques, we normalise all foreign words in the Arabic text using rules that bring similar words together. Techniques such as Soundex and Phonix, described in Section 2.2.3, normalise words by replacing characters with codes based on their phonemes. Words in the query are similarly converted into phonetic forms for lookup in the index. In dynamic techniques, words in the query are compared to words in the index at search time; the similarity between two words is estimated using techniques such as n-grams, Edit Distance, or Editex (described in Section 2.2.3).

7.2.1 Static Algorithms

We propose two new algorithms that deal with foreign words at indexing time: NORM and Soutex.

The NORM Algorithm

Our first algorithm to deal with foreign word variants is called "NORM". This normalises words by removing vowels and keeping the first and the last characters, replacing transliterated characters that originate from one English character to a single Arabic character; we consider diphthongs and double vowels in this mapping. To develop this algorithm we run different versions and test them on the first data set. Table 7.2 shows the different versions and their descriptions.

In our initial version (NORM1), we only remove vowels from every foreign term. In the second version (NORM2), we keep vowels unchanged if they are the first or the last characters of the word, since they are generally pronounced in Arabic. The long vowel letters are sometimes used as consonants, and these may be followed immediately by another long vowel. For example, the vowel letter"ي" /j/ may be followed by the long vowel"و" /w/ to form "يو" /jw/. For such cases, we keep the first vowel and remove the second. Two vowels can also be used together as diphthongs, as in "او" /aw/ and "اي" /aj/, where the diphthongs are caused by not vocalising the second vowel. We retain vowels that are followed by another vowel or preceded by a vowel that forms a diphthong. This forms the third version of our algorithm (NORM3). We also conflate similar consonants based on

Algorithm	Description
NORM1	Remove all vowels
NORM2	NORM1 + Do not remove vowels at position 1 and n in an n-character word
NORM3	NORM2 + Keep vowels if they are followed by another vowel or form a diphthong
NORM	NORM3 + Replace characters originated from the same English character with one character

Table 7.2: NORM algorithm development.

Original	Normalised
س ش ز ص	س
ت ط	ط
ج غ ك ق	غ
ث	ت

Table 7.3: Normalisation of equivalent consonants to a single form.

statistical analysis of letter mappings between English and Arabic [Abduljaleel and Larkey, 2003; Stalls and Knight, 1998], and confirming through a web search that these consonants are used interchangeably in web documents. Table 7.3 shows all consonants we consider to be equivalent. Our process may lead to ambiguity where a similar native word exists; for instance, the spelling variants ايلاند and ايلند for ⟨island⟩ are normalised to الند, which is identical to the Arabic word (/annid̪/⟨equivalent⟩). Adding a custom prefix (not found in Arabic text) to the normalised form is one way to address this issue; we choose to add the letter "ة" to the beginning of each normalised word. For example, variants for "island" are thus normalised to ةالند. Since the letter "ة" never occurs at the beginning of any Arabic word, no ambiguity remains.

To evaluate the effectiveness of our approaches, we consider each word in the list to be a query, and pose this to the entire collection. The query result should be other words in the same cluster. We measure the effectiveness using average precision and average recall over all queries.

Characters	Code
ا و ي	0
ة ت ط ث ظ ض	1
س ش ز ص	2
د ذ	3
ج غ ك ق	4
ع ه ح	5
ن	6
م	7
ف	8
ل	9
ب	A
ر	B
خ	C

Table 7.4: Mappings for our phonetic approach.

The Soutex Algorithm

Using the letter groups identified on the previous section, we also developed an algorithm similar to Soundex to conflate transliterated foreign words in Arabic. We did not consider all sounds in Arabic; instead, we addressed only those sounds that are found in transliterated foreign words. We group sounds based on statistical analysis of letter mappings between English and Arabic [Abduljaleel and Larkey, 2003; Stalls and Knight, 1998], and after using the Google search engine to confirm that these consonants are used interchangeably in practice. For example, a search for the transliteration variants غورباتشوف, قورباتشوف, جورباتشوف, and كورباتشوف for "Gorbachev" confirmed that the English sound /g/ can be mapped to جـ /ʒ/, غـ /ɣ/, قـ /q/, or كـ /k/ in Arabic, and so we map these letters to the same code 4. Our phonetic algorithm aims to replace similar transliterated sounds with a single code. As noted earlier, we do not envisage that this algorithm has use for native Arabic words, as these are usually distinct, and pronunciation is rarely ambiguous. Table 7.4 shows Arabic letters and their corresponding codes. To normalise a foreign word, we replace each letter but the first by its phonetic code, and drop any vowels. We call this version "Soutex".[1] In this version,

[1]This name is a play on the Arabic word صوت (/sˤwt̪/⟨sound⟩).

we do not limit encoding to a specific number of characters as it has been empirically shown that this is neither effective for English [Zobel and Dart, 1996] nor for Arabic [Aqeel et al., 2006]. However, as our task is different, we also test the effectiveness of limiting encoding to four characters as in the English Soundex. Therefore, we use another version in which we only encode the first four characters in the word. We call this version "Soutex4".

7.2.2 Dynamic Algorithms

We apply most of the string similarity techniques discussed in Seacion 2.2.3 to Arabic and check their effectiveness in capturing variants of foreign words. We specifically test the gram count (gramCount), gram distance (gramDist), dice (Dice), edit distance (Edit Distance), longest common subsequence (LCS), and skip grams (Sgrams). We also extend the Editex technique to Arabic by replacing the character groups used for English with Arabic character groups. We then modify this technique and improve its ranking. In this thesis, we use the term "dynamic algorithms" when referring to only the algorithms listed here, and do not imply that our conclusions apply to dynamic algorithms in general.

Arabic Editex

Based on groups identified in Table 7.5, we have modified the Editex algorithm of Zobel and Dart [1996] explained in Section 2.2.3. This works in the same manner as in English except that we drop the functionality used to consider the two silent characters in the English version, since silent characters in Arabic are rare and usually occur at the beginning or at the end of the word. More specifically, we replace $d(s_i, t_j)$ by $r(s_i, t_j)$. We call the Arabic version of this algorithm "AEditex". The distance between two strings s and t is computed as:

$$\begin{aligned}
&edit(0,0) = 0\\
&edit(i,0) = edit(i-1,0) + d(s_i - 1, s_1)\\
&edit(0.j) = edit(0,j-1) + d(t_j - 1, t_j)\\
&edit(i.j) = min[edit(i-1,j) + d(s_i - 1, s_i),\\
&\qquad\qquad edit(i,j-1) + r(s_i, t_j),\\
&\qquad\qquad edit(i-1,j-1) + r(s_i, t_j)]
\end{aligned} \tag{7.1}$$

where $r(s_i, t_j)$ is 0 if s_i=t_j, 1 if $group(s_i)$=$group(t_j)$, and 2 otherwise.

Characters	Group
ا و ي	0
ت ث	1
ت ط	2
ظ ض	3
س ش	4
س ص	5
ز س	6
د ذ	7
ج غ ك ق	8

Table 7.5: AEditex letter groups.

Ranked AEditex

Edit Distance ranks words by the number of steps required to transpose one word to another. This generates integer ranks, and so many words may have the same rank. For example, given the word ايثيلين ⟨Ethylene⟩ as a query, Edit Distance ranks the words ايتيلن ⟨a variant of Ethylene⟩, and ايفيلين ⟨Evelynne⟩ equally, as only one step is needed to change each one to the query word. The word ايتيلن is a variant of the query, and differs only in spelling; the other word however, differs in both spelling and pronunciation. AEditex resolves this problem by grouping similar sounds and assigning words with similar pronunciation lower distance than those with same distance but having different pronunciation. AEditex, however, still produces weak ordered ranks, and more fine-grained ranking may improve results.

To differentiate between words and to reduce the size of ranks, we introduce the concept of real-valued distance. In AEditex, words with the same characters have a distance of zero, words with one different character have a distance of two, and words with only one different character that is similar in pronunciation to its counterpart in the second word have a distance of 1. AEditex thus has two ranks for cases where characters are not identical. We believe that the rank of words with different characters but similar pronunciation can be further improved.

Consider the two pairs "جابِرِيل" /ɣabirjil/ and "غابِرِيل" /ɣabirjil/ ⟨transliterations of the proper noun "Gabriel"⟩, and "تونِي" /t̪wnj/ and "طونِي" /t̪ˤwnj/ ⟨transliterations of the proper noun "Tony"⟩. Using AEditex, the similarity between the first pair is equal to the

AEditex Ranking				REditex Ranking			
Word	Distance	Words	Distance	Word	Distance	Word	Distance
ايثيلين	0	ايلين	4	ايثيلين	0.00	ويتولين	1.83
ايتيلين	1	اصيلين	4	ايتيلين	0.50	تمثيليّن	2.00
اثيلين	2	ايثيل	4	اثيلين	0.66	ايزيليو	2.00
ايثيلن	2	ايميليّه	4	ايثلين	1.00	فيسيلين	2.00
ايفيلين	2	اثلين	4	ايثيلن	1.00	ايميليّه	2.00
ايثلين	2	اثيرين	4	ايفيلين	1.00	ايثيل	2.00
اتيلين	3	تمثيليّن	4	اتيلين	1.17	انجيليّن	2.00
ايميلي	4	ايفلين	4	اصيلين	1.67	ايزيديّن	2.00
ليويلين	4	فيسيلين	4	ايثيقيون	1.67	ايريلي	2.00
ايثيقيون	4	ايطاليّن	4	ايطاليّن	1.67	ايفلين	2.00
ايريلي	4	ايزيليو	4	ايلين	1.67	ايرلين	2.00
ايزيديّن	4	انجيليّن	4	اثيرين	1.67	ليويلين	2.00
ايرلين	4	انثيّن	4	اثلين	1.67	انثيّن	2.00

Table 7.6: Comparison of AEditex and REditex ranking. Words retrieved as variants for the word "ايثيلين". Words are ranked based on values of both AEditex, and REditex.

second pair. With the phonetic groups used in AEditex, the probability of transliterating the source character "G" to "ج" or "غ" is $\frac{1}{4}$, whereas the probability of transliterating the character "T" to "ت" or "ط" is $\frac{1}{2}$. Based on this, we introduce another new rule to AEditex. If two characters are the same, the function $r(s_i, t_j)$ returns 0, if they are not the same but belong to the same group, it returns $1 - \frac{1}{\textit{the group length}}$, and if they are not the same and do not belong to the same group, it returns 1. Using groups identified in Table 7.5 the probability is either $1 - \frac{1}{2}$, $1 - \frac{1}{3}$, or $1 - \frac{1}{4}$. Under this scheme, the similarity between the first pair is 0.75, while the similarity between the second pair is 0.50. We believe that this is more realistic than with AEditex. We call our modified algorithm "REditex".

Table 7.6 shows an example of ranking the seven variants of the word "ايثيلين" ("ايثيلين", "ايتيلين", "اثيلين", "ايثيلن", "ايثلين", "اتيلين", and "اثلين") among words retrieved using AEditex and REditex algorithms. Both algorithms retrieve the seven variants, but REditex produces much better ranking. The last rank in AEditex (Rank 4) is divided by REditex into three ranks.

			ا	ي	ت	ي	ل	ي	ن
			a	y	**t**	y	l	y	n
		0	2	4	6	8	10	12	14
ا	a	2	0	2	4	6	8	10	12
ي	y	4	2	0	2	4	6	8	10
ث	th	6	4	2	**1**	3	5	7	9
ي	y	8	6	4	3	1	3	5	7
ل	l	10	8	6	5	3	1	3	5
ي	y	12	10	8	7	5	3	1	3
ن	n	14	12	10	9	7	5	3	<u>1</u>

		ا	ي	ت	ي	ل	ي	ن
		a	y	**t**	y	l	y	n
		0.00	1.00	1.67	2.67	3.67	4.67	5.67	6.67
ا	a	1.00	0.00	0.67	1.67	2.67	3.67	4.67	5.67
ي	y	1.67	0.67	0.00	1.00	1.67	2.67	3.67	4.67
ث	th	2.67	1.67	1.00	**0.50**	1.50	2.50	3.50	4.50
ي	y	3.67	2.67	1.67	1.50	0.50	1.50	2.50	3.50
ل	l	4.67	3.67	2.67	2.50	1.50	0.50	1.50	2.50
ي	y	5.67	4.67	3.67	3.50	2.50	1.50	0.50	1.50
ن	n	6.67	5.67	4.67	4.50	3.50	2.50	1.50	<u>0.50</u>

Figure 7.1: Calculating Editex: AEditex (left) and REditex (right). Change in results is indicated in bold and the final distance is underlined.

REditex ranks the last variant "اثلين" in the sixth position in rank 4. As this is a weak rank — all words have distance value 1.67 — that starts at the eighth position and ends at the thirteenth position, this result is the worst possible result that REditex can produce. In contrast, AEditex ranks the same variant at the 18th position, with the possibility that this variant falls in the 26th position. Using the probability of relevance measure (PRR) described in Section 2.3.3, the precision of AEditex is 0.412, while for REditex it is 0.667.

Figure 7.1 shows how AEditex and REditex are calculated. Latin characters are used to represent the two words involved in the calculation. Both algorithms follow the same strategy in comparing the two words. They only differ when reaching position (3,3) at which the two characters are not the same but belong to a two-letter group. REditex returns 0.5 while AEditex returns 2. Since all other letters are the same, this is the final distance.

7.3 Evaluation

As discussed in Section 2.3.3, results returned by the static algorithms and the dynamic algorithms discussed in the past section are not directly comparable, as dynamic algorithms return ranked results and static ones return unranked results. Both techniques result in a weak-ordered ranking. As such, in this section we use the PRR measure described in Section 2.3.3 to compare these approaches. We present results on the crawled and the transliterated data sets in the recall-precision graph over the 11-recall points.

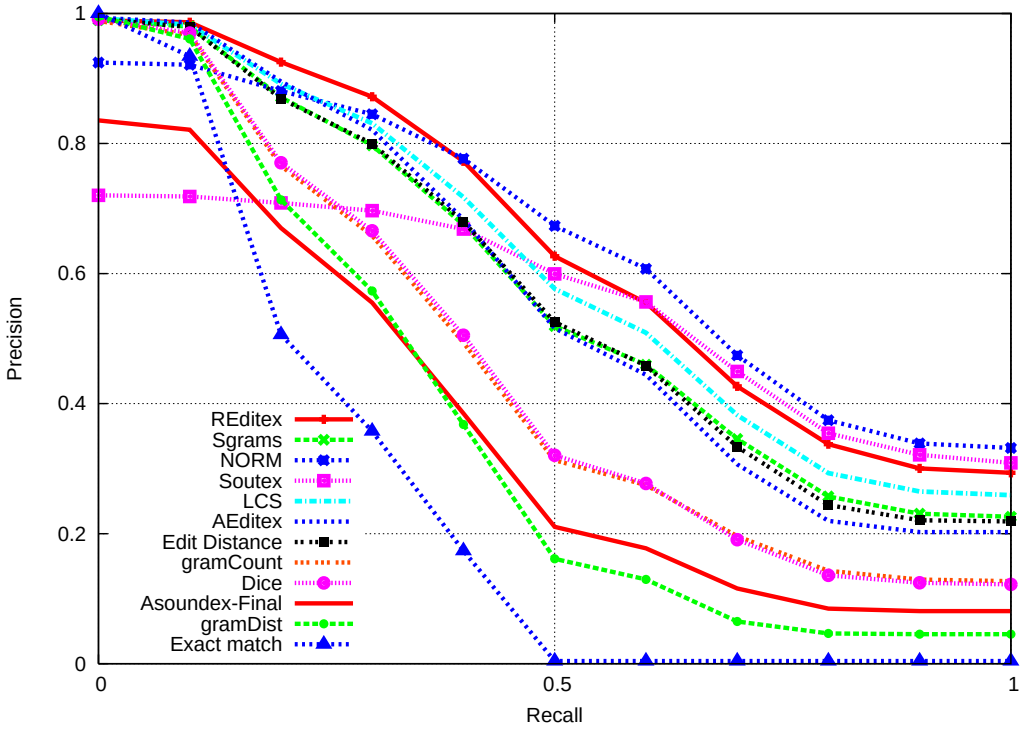

Figure 7.2: Results of static and dynamic algorithm on the crawled data.

7.3.1 Results and Discussion

Results obtained from running algorithms using queries in both data sets against their respective collection are shown in Figure 7.2 and Figure 7.3. The average precision (average PRR in our case) for each algorithm is shown in Table 7.7. Algorithms produce results that are significantly better than exact match [t-test, $p < 0.001$].

On the first data set, NORM performs the best. REditex is the second-best algorithm, followed by LCS, AEditex and Edit Distance. Soutex shows better performance than all other algorithms except NORM after 50% recall, but performs poorly at lower recall levels. Both the gramCount and Dice algorithms have similar performance with average precision at around 46%. Asoundex-final and gramDist show poorer performance than other algorithms, with average precision at 38%.

Results from the transliterated data set generally favour the string similarity algorithms. REditex outperforms all other techniques with an average precision of 82%, followed by LCS at 78%, Sgrams at 76%, Edit Distance at 70%, and then AEditex at 62%. Soutex performs

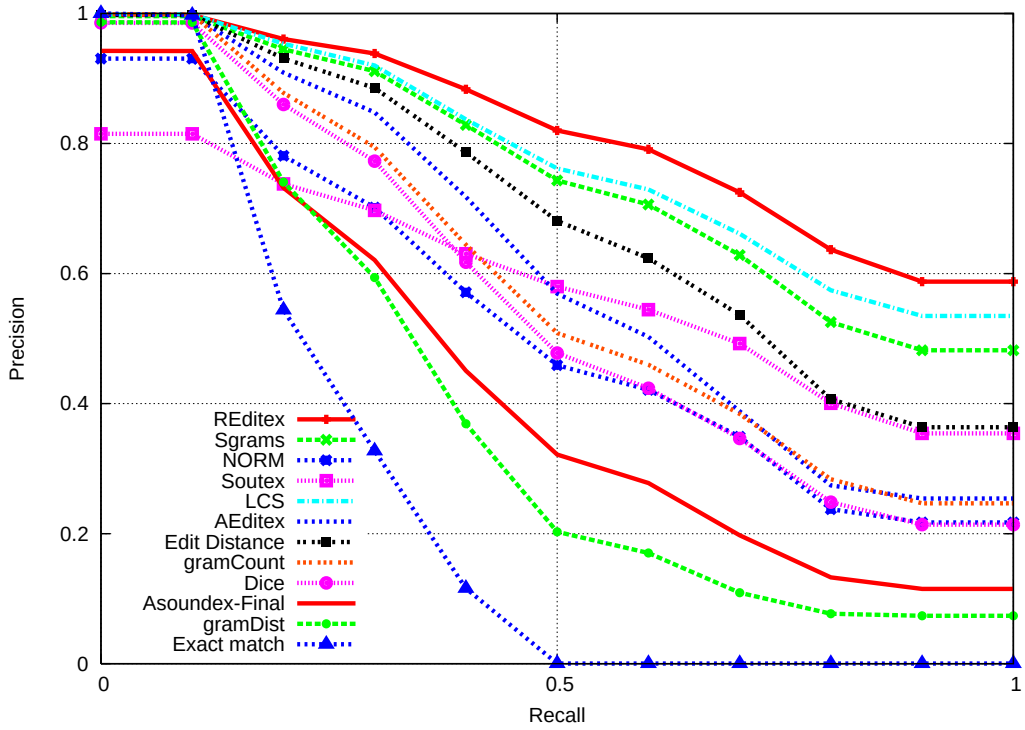

Figure 7.3: Results of static and dynamic algorithm on the transliterated data.

better than both the gramCount and Dice algorithms. It also performs better than AEditex at recall levels of 50% and higher. NORM performs better than the Asoundex-final and gramDist algorithms. The gramDist algorithm is again the worst. All algorithms showed significant improvement above the baseline [t-test, $p < 0.001$].

Although the NORM and Soutex algorithms do not produce the best performance, they have the advantage of generating encodings for later use in retrieval. Dynamic algorithms are more computationally expensive and can only be used at query time. In the next section we show how these algorithms can be used in a real IR environment.

7.4 IR Evaluation

In this section we use the above algorithms to find foreign words in Arabic text. Algorithms classified as static are easily implemented and can be integrated with any AIR systems when processing text for indexing. However, algorithms classified as dynamic are more difficult to integrate into AIR systems, they need to be run concurrently as the user types a query to

	Data set	
Algorithm	First	Second
Exact Match	0.300	0.261
REditex	0.656	**0.820**
LCS	0.619	0.782
Sgrams	0.586	0.759
Edit Distance	0.572	0.700
AEditex	0.576	0.624
NORM1	0.548	0.534
NORM2	0.575	0.463
NORM3	0.549	0.459
NORM	0.660	0.536
Soutex	0.530	0.590
gramCount	0.451	0.595
Dice	0.457	0.568
Asoundex-final	0.368	0.446
gramDist	0.376	0.401

Table 7.7: Average precision for all algorithms. All show significant improvement over the baseline with REditex performs the best. Exact Match is the baseline.

compare words in the query with words in the collection index.

7.4.1 Experimental Setup

With dynamic algorithms, foreign words in the user's query are compared at query time to words in the collection index. We can decide whether a query word is sufficiently similar — using a threshold that we empirically determine — to a word in the index to warrant replacement of the query word with the corresponding word that appears in the index.

We use the AGW collection with 90 queries along with their relevance judgements. Most queries (64 of 90) contain foreign words. To minimise the time required to check words in the collection against foreign words in the query, we use the most effective identification technique presented in the past chapter (N-grams with cutoff value 0, combined with the Microsoft Office 2003 lexicon) to filter out foreign words from both the collection and the queries. This step resulted in identification of 64 unique foreign words in the query title,

description, and narrative fields. Similarly, of the 2 209 850 unique words in the collection, we determined 594 139 of these (26.9%) to be foreign. By applying algorithms to only the foreign words, we achieve two objectives: first, we avoid applying algorithms specifically developed for foreign words to Arabic words, which might cause Arabic words to be reformed and indexed under the wrong reference term in the index. Second, we limit comparison of words in the query to 26.9% of the words in the collection rather than comparing with all the words in the collection, representing substantial efficiency gain for dynamic algorithms.

We run both static and dynamic algorithms to search the collection for variants of foreign words appearing in the query. If a word is judged as a variant, we replace it with the variant of the word found in the query. In such cases, all identified variants in the collection will be replaced with the same variant.

As dynamic algorithms return a ranked list of variants with the best match at the top, we run every algorithm with its different possible thresholds starting at the top rank and increasing the threshold to gradually include other ranks. We report the best result for every algorithm with its respective threshold. We have determined that for this task, the best result is usually achieved when using variants returned at the top rank.

We use the light11 stemmer to stem both the collection and the queries. We extend the stemmer with our algorithms to return the appropriate version of the word if it is found in the list of filtered foreign words. This stemmer is used as it was the best variant demonstrated in Chapter 5.

The light11 algorithm starts by normalising words, then removes the particle "و", the definite article, and suffixes. We check whether a word is foreign after the second step — after removing the particle "و". Figure 7.4 shows how both static and dynamic algorithms work with the light11 stemmer. When using a static algorithm, a word is encoded only if it is a foreign word. In contrast, when using a dynamic algorithm the version of word in the query is used to replace words in the collection found to be sufficiently similar to it. We use the Okapi BM25 weighting scheme with the best values that we determined in Chapter 5 (b=0.25, K_1=1, and K_3=7). We did not use any relevance feedback technique in this experiment.

7.4.2 IR Results

Table 7.8 shows results of indexing the collection using static and dynamic algorithms. None of the algorithms add significant improvement to the light11 stemmer when using the MAP measure. NORM1, NORM, and AEditex algorithms have the best improvement in MAP,

Require: $length(w) > 0$
  $w \Leftarrow RemovePunctuation(w)$
  $w \Leftarrow Normalise(w)$
  if $w[i]$ =' و' **then**
    $w \Leftarrow copy(w, 2, length(w) - 1)$
  end if
  if $IsAForeignWord(w)$ **then**
    $w \Leftarrow encode(w)$
    return w
  end if
  $w \Leftarrow RemoveAlPrefixes(w)$
  $w \Leftarrow RemoveSuffixes(w)$
  return w

(a) Static algorithms within light11.

Require: $length(w) > 0$
  $w \Leftarrow RemovePunctuation(w)$
  $w \Leftarrow Normalise(w)$
  if $w[i]$ = ' و' **then**
    $w \Leftarrow copy(w, 2, length(w) - 1)$
  end if
  if $IsAForeignWord(w)$ **then**
    for $i = 1$ to NoFWinQuery **do**
      if (dynamic(w,FWinQuery[i]) *lop* threshold) **then**
        return FWinQuery[i]
      end if
    end for
  end if
  $w \Leftarrow RemoveAlPrefixes(w)$
  $w \Leftarrow RemoveSuffixes(w)$
  return w

(b) Dynamic algorithms within light11.

Figure 7.4: Static and dynamic algorithms integrated within the light11 stemmer: "encode" represents a static algorithm; "dynamic" represents a dynamic algorithm; "lop" represents a logical operator and is either ">, <, =, >=, or <="; and $IsAForeignWord(w)$ is a function that searches a word w in the identified foreign words list.

but this is only weakly significant [t-test, $p = 0.078$, $p = 0.079$, and $p = 0.075$ respectively]. NORM1 improves the MAP measure by 9.64%, followed by AEditex (8.96%), NORM (8.53%), and NORM3 (7.93%). Recall also increases significantly with NORM1, NORM and AEditex [t-test, $p = 0.043$, $p = 0.020$, and $p = 0.041$ respectively]. NORM2, and NORM3 improve recall, but improvement is only weakly significant [t-test, $p = 0.070$, and $p = 0.072$ respectively].

The phonetic algorithms, along with gramDist, decrease the performance of the light11 stemmer significantly in all measures.

The AEditex algorithm adds the second best increase to the light11 stemmer, resulting in a weakly significant improvement in P@10 [t-test, $p = 0.063$]. In contrast to our previous results with the list of foreign words variants, integrating AEditex in the light11 algorithm

Technique	Threshold	MAP	P@10	RP	RECALL
light11	-	0.2053	0.2978	0.2378	0.6456
wNORM1	-	**0.2251**	0.3200	0.2508	**0.6791**↑
wNORM2	-	0.2217	0.3111	0.2467	0.6655↑
wNORM3	-	0.2216	0.3111	0.2471	0.6652↑
wNORM	-	0.2228	0.3111	0.2527	0.6712↑
wSoutex4	-	0.1639↓	0.2433↓	0.1930↓	0.6008↓
wSoutex	-	0.1630↓	0.2433↓	0.1930↓	0.5981↓
wAsoundex-Final	-	0.1492↓	0.2233↓	0.1742↓	0.5593↓
wDice	> 0.5	0.2049	0.2911	0.2327	0.6496
wgramCount	> 0.8	0.2063	0.2933	0.2329	0.6518
wgramDist	≤ 2.0	0.1275↓	0.1922↓	0.1544↓	0.5213↓
wSgrams	> 0.8	0.2052	0.2922	0.2325	0.6496
wLCS	> 0.8	0.2083	0.2967	0.2345	0.6511
wEditDistance	≤ 1.0	0.2066	0.2922	0.2337	0.6508
wAEditex	< 3.0	0.2237	**0.3244**	**0.2539**	↑0.6617
wREditex	≤ 1.0	0.2058	0.2911	0.2334	0.6506

Table 7.8: Performance of light11 stemmer with our static and dynamic algorithms. AEditex and NORM algorithms produce the best results. ↑ indicates values that are significantly better than the light11 stemmer at the 95% confidence level, while ↓ indicates values that are significantly worse than the light11 stemmer.

outperformed the integration of REditex. It is significantly better than REditex in MAP [t-test, $p = 0.058$], P@10 [t-test, $p = 0.006$], and R-Precision [t-test, $p = 0.038$]. It is also significantly better than integrating the Edit Distance algorithm in P@10 and R-Precision [t-test, $p = 0.008$, and $p = 0.041$ respectively].

Comparing the NORM algorithms with REditex, only NORM adds significant improvement in both recall and R-Precision [t-test, $p = 0.027$, and $p = 0.044$ respectively]. NORM1, NORM2, and NORM3 add only weakly significant gains over REditex. The performance of the best-performing algorithms is shown in Figure 7.5.

To investigate the effects of our introduced algorithms in more detail, we show retrieval results for individual queries. Due to the large number of queries, we only show those affected by incorporating our algorithm (NORM) into the light11 stemmer. If the absolute

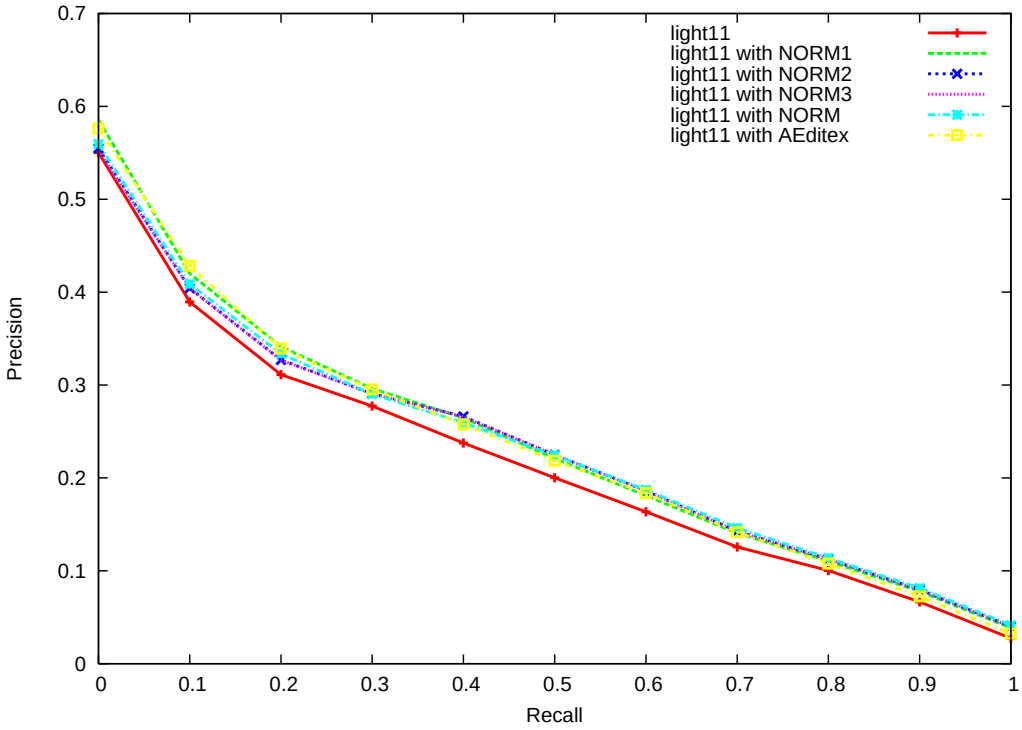

Figure 7.5: The effects of foreign word normalisation on the light11 stemmer using the NORM and AEditex algorithms. Algorithms perform equally and enhance the light11 retrieval performance.

value of the difference between the average precision before and after integrating NORM in the light11 stemmer is less than 0.01, we exclude the query. Figure 7.6 shows the effects of our NORM algorithm on the light11 stemmer with performance measured by average precision. The graph shows that 21 queries have been improved by adding the NORM algorithm; the increase is quite marked for some queries; for example, queries 8, 51, and 84 achieve 0 in MAP when using the light11 stemmer alone, but score 0.0174, 0.3540 and 0.1232 respectively when integrating the NORM algorithm. Similarly we observe 0.4762, 0, and 0 Recall when using the light11 alone, but score 0.7619, 0.7778, and 0.8571 respectively when applying the NORM algorithm. This is due to the fact that using the light11 stemmer alone failed to conflate foreign word variants in the document collection with the variants of foreign words used in the queries. For example, Query 44 "اعصار سنلاكو بالصين" ⟨the typhoon Sinlaku in China⟩, scores a MAP of 0.3403 when using the light11 stemmer alone, but scores 1.000 when applying the NORM algorithm. There are 7 documents relevant to this query. Recall

is 100% in both cases, but the ranking is different. The light11 stemmer alone ranks only 4 documents within the top 10 retrieved documents (P@10=0.4000). These documents contain the same query variant "سنلاكو" (/snlakw/⟨the typhoon Sinlaku⟩). The other relevant documents that contain the second variant "سينلاكو" /sinlakw/ are ranked after the top 30 retrieved documents (P@30=$\frac{4}{30}$=0.1333), with the last relevant document retrieved beyond the top 200 retrieved documents (P@200=$\frac{6}{200}$=0.0300). Applying the NORM algorithm results in ranking all 7 relevant documents at the top 10 retrieved documents (P@10=0.7000), indicating that the two variants are conflated together.

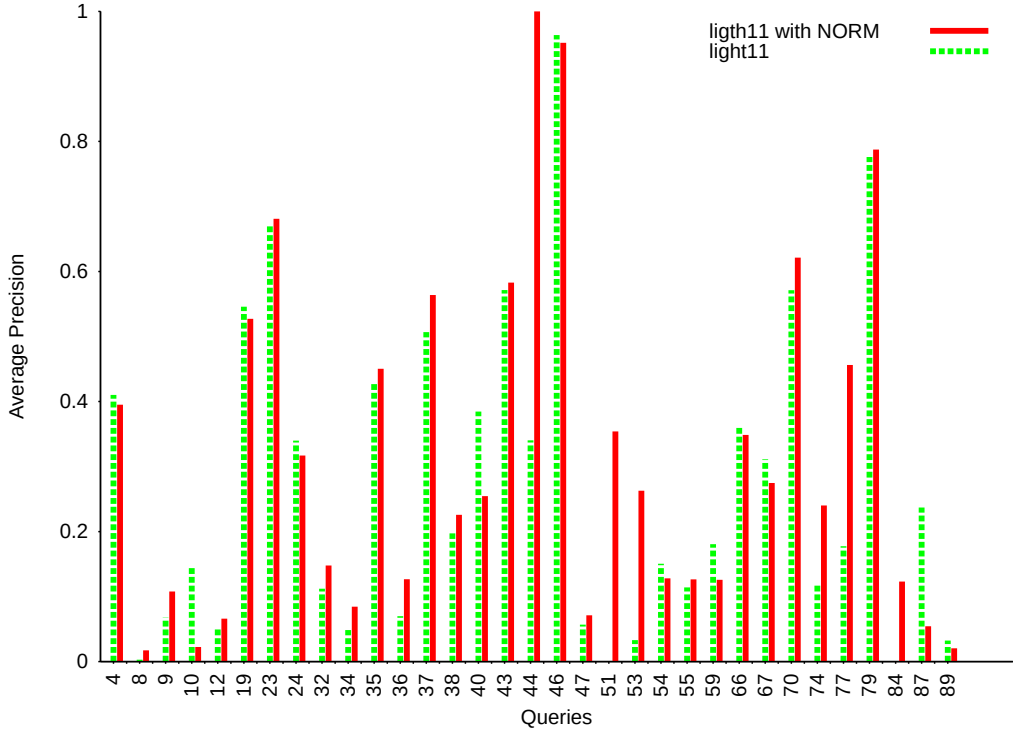

Figure 7.6: Queries affected by the integration of the NORM algorithm in the light11 stemmer. 21 queries are positively affected, while 12 are negatively affected. Improvement is more substantial than loss.

Despite the improvement that the NORM algorithm has on some queries, it negatively affects 12 other queries. Queries 10, 40, and 87 are the most affected.

7.4.3 Using Query Expansion

In this section we test query expansion by replacing the original foreign word in the query by different variants returned by the different algorithms.

We use the INQUERY's structured query language [Callan et al., 1995] to expand foreign words with their variants. The INQUERY retrieval method accepts a query and returns a belief list that contains a list of documents and their corresponding probabilities of satisfying the query. The query is structured using several operators that determine the final belief, using beliefs generated from different terms in the query [Callan et al., 1992].

We first convert queries (titles only) by applying the #sum operator to include all terms in the query, then we expand foreign words by enclosing all variants returned by individual

	MAP	P@10	RP	RECALL
light11	0.1736	0.2533	0.2003	0.6102

Table 7.9: Results of running the light11 stemmer on the AGW collection using the INQUERY retrieval model. Query terms are grouped using the operator #sum.

algorithms within the #syn operator. This allows variants to contribute equally to the belief of the foreign word in the query. The final query belief is generated by the #sum operator which calculates the mean of beliefs of all terms in the query. An example of an expanded query is #sum(بالصين #syn(سنلاكو سينلاكو) اعصار). In this example, the word سنلاكو is expanded with two variants.

Our main objective in this section is to test the effects of query expansion using the different variants of a foreign word. As the retrieval model is different from the one used previously (Okapi BM25), scores reported in this section are not directly comparable with the previous ones. Table 7.9 shows the baseline results using the INQUERY retrieval model, running the light11 stemmer without any expansion.

To expand queries using variants returned by different algorithms, there are two main issues that need to be considered: first, the number of variants used to expand the query; and second, the process of choosing variants from the returned unranked lists.

The algorithms return different number of variants, with the phonetic similarity algorithms generally returning fewer variants than the string similarity algorithms. Using a fixed number of variants might favour one algorithm over another. Therefore, we use different numbers of variants, starting with as few as three variants up to 100 returned variants.

The second issue is related to selecting variants from unranked lists such as those returned by Soutex, and the NORM algorithms. To overcome this issue, we rank variants in unranked lists using the Dice measure (Section 2.2.3). This approach has been used by Holmes and McCabe [2002] to overcome the problem of evaluating weak-ranked results returned by the Soundex algorithm. We rank variants returned by the Soutex, Soutex4, Asoundex-Final, NORM1, NORM2, and NORM3 algorithms based on their similarity with the foreign word in the query. After ranking, we choose the first n variants to replace the foreign word in the query within the #syn operator. We test the expansion using the top 3, 5, 10, 20, 30, 40, 50, and 100 ranked variants.

To test the effects of expanding all foreign words in queries and not only those identified by our identification algorithm, we have manually inspected the AGW topics and identified

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM1	0.1785	0.1843	0.1876	0.1927↑	0.1910	0.1891	0.1890	0.1893
NORM2	0.1791	0.1853	0.1845	0.1844	0.1844	0.1844	0.1844	0.1844
NORM3	0.1791	0.1853	0.1845	0.1844	0.1844	0.1844	0.1844	0.1844
NROM	**0.1821**	**0.1874**	**0.1886**	0.1874	0.1883	0.1883	0.1883	0.1883
Soutex4	0.1771	0.1811	0.1845	**0.1939**↑	**0.1924**↑	**0.1924**↑	**0.1965**↑	**0.1927**
Soutex	0.1758	0.1758	0.1821	0.1857	0.1851	0.1840	0.1840	0.1823
Asoundex-Final	0.1820	0.1827	0.1831	0.1827	0.1827	0.1827	0.1827	0.1827
Dice	0.1712	0.1690	0.1685	0.1709	0.1738	0.1711	0.1698	0.1612
gramCount	0.1716	0.1690	0.1685	0.1710	0.1739	0.1731	0.1719	0.1722
gramDist	0.1646	0.1633	0.1619	0.1710	0.1673	0.1655	0.1622	0.1558
Sgrams	0.1624	0.1624	0.1650	0.1690	0.1696	0.1689	0.1682	0.1656
LCS	0.1722	0.1761	0.1824	0.1825	0.1811	0.1869	0.1864	0.1830
EditDistance	0.1655	0.1764	0.1800	0.1789	0.1809	0.1844	0.1830	0.1809
AEditex	0.1677	0.1760	0.1832	0.1780	0.1828	0.1839	0.1829	0.1786
REditex	0.1770	0.1841	0.1827	0.1828	0.1813	0.1841	0.1840	0.1801

Table 7.10: The MAP scores of the light11 stemmer when expanding queries using the first 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. Soutex4 adds significant improvement over the non-expanded baseline (MAP=0.1736). Foreign words expanded are only those automatically identified as foreign in the queries. ↑ indicates values that are significantly better than the light11 stemmer at the 95% confidence level.

114 foreign words, 50 more than the 64 detected by the foreign word identification algorithm described in Section 6.4.2. We experimented with both foreign word sets. Having 15 different algorithms and 8 different expansion sets for both manually and automatically identified foreign words, we have 240 different runs in total. In each run, we stemmed the queries using the light11 stemmer, expanded foreign words in queries using the appropriate number of variants, and ran them against the collection index. Results of expanding the automatic identified foreign words in the AGW queries are shown in Table 7.10 and those returned by expanding all manually identified foreign words are shown in Table 7.11. We show only the MAP measure. Results for other measures are shown in Appendix B.

Two algorithms result in a significant increase in MAP. These are the NORM1 and the Soutex4 algorithms. The increase that the NORM1 algorithm adds is only significant when using the top 20 variants [t-test, $p = 0.039$], and weakly significant when using the top 30, 40,

and 100 variants [t-test, $p = 0.061$, $p = 0.097$, and $p = 0.093$ respectively]. Soutex4 adds significant improvement when expanding queries with the top 20, 30, 40, and 50 variants [t-test, $p = 0.030$, $p = 0.042$, $p = 0.041$, and $p = 0.022$ respectively]. When expanding the queries with the top 100 variants, the improvement is significant at the 94% confidence level [t-test, $p = 0.055$]. The Soutex algorithm results in a weakly significant improvement when using the top 20 [t-test, $p = 0.055$], 30 [t-test, $p = 0.068$], 40 [t-test, $p = 0.091$], and 50 [t-test, $p = 0.085$].

Phonetic similarity algorithms retrieve fewer variants. For example, NORM2, NORM3, and Asoundex-Final return less than 10 variants, while the Soutex algorithm returns up to 30 variants. Soutex4 benefited from the large number of variants and the ordering of these variants using the Dice algorithm.

Although Sgrams and gramDist reduce the performance of the light11 stemmer at all expansion levels, the decrease is only weakly significant when using the top 3 and 5 variants of the Sgram algorithm [t-test, $p = 0.060$, and $p = 0.066$ respectively], and the top 10 variants of the gramDist algorithm [t-test, $p = 0.062$].

Considering the performance of the same algorithms when expanding all manually identified foreign words in the queries, none add a significant improvement to the light11 stemmer. In fact, results are worse than using the automatic expansion. We relate this to the vagueness of some words identified as foreign. Humans rely on context to determine whether a word is foreign. As explained in Section 2.1.5, a foreign word may be spelt identically to a native Arabic word, but with different (normally omitted) diacritics. Moreover, our identification algorithms avoid classifying words that have three or fewer characters. In most cases, such words are interpreted differently. For example, the words "أي" (/ʔj/⟨which⟩), "بام" (/biʔumm/⟨with the mother of⟩), and "بول" (/bawl/⟨urine⟩) are in fact foreign words with the meaning "A", "BAM", and "Paul" respectively. In general, the phonetic similarity algorithms outperform string similarity algorithms in both experiments.

7.5 Chapter Summary

Foreign words transliterated into Arabic can appear with multiple spellings, hindering effective recall in a text-retrieval system. We have examined nine techniques to find such variants. Edit Distance, Gram Count, Dice, Gram Distance, and Longest Common Subsequence are language-independent techniques used to find variant names in other languages; Asoundex-Final, Soutex, AEditex, and REditex are extended techniques to accommodate

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM1	0.1733	**0.1824**	**0.1878**	**0.1886**	0.1868	0.1844	0.1838	**0.1817**
NORM2	0.1699	0.1720	0.1700	0.1702	0.1702	0.1702	0.1702	0.1702
NORM3	0.1687	0.1707	0.1687	0.1689	0.1689	0.1689	0.1689	0.1689
NORM	0.1702	0.1720	0.1719	0.1713	0.1715	0.1711	0.1703	0.1705
Soutex4	0.1665	0.1708	0.1826	0.1885	**0.1901**	**0.1891**	**0.1849**	0.1769
Soutex	0.1708	0.1747	0.1788	0.1819	0.1816	0.1813	0.1821	0.1799
Asoundex-Final	0.1681	0.1675	0.1654	0.1652	0.1641	0.1639	0.1638	0.1638
Dice	0.1673	0.1648	0.1648	0.1686	0.1717	0.1690	0.1663	0.1568
gramCount	0.1664	0.1639	0.1684	0.1684	0.1728	0.1685	0.1678	0.1658
gramDist	0.1635	0.1598↓	0.1664	0.1743	0.1658	0.1643	0.1606	0.1556
Sgrams	0.1550↓	0.1549↓	0.1577↓	0.1639↓	0.1616↓	0.1623↓	0.1627↓	0.1586↓
LCS	0.1715	0.1753	0.1822	0.1816	0.1800	0.1857	0.1851	0.1784
EditDistance	0.1738	0.1738	0.1738	0.1738	0.1738	0.1738	0.1738	0.1738
AEditex	0.1614	0.1723	0.1822	0.1757	0.1799	0.1817	0.1809	0.1740
REditex	**0.1740**	**0.1824**	0.1831	0.1831	0.1801	0.1829	0.1834	0.1767

Table 7.11: The MAP scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. Soutex4 adds significant improvement to the non-expanded baseline (MAP=0.1736). Foreign words expanded are those manually identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer.

Arabic Words; and NORM is a novel technique to find foreign word variants in Arabic. We have shown that these techniques are effective in finding foreign word variants.

We have developed different versions of the NORM algorithm to normalise foreign words in Arabic. We first remove vowels from foreign words, keeping the first and last characters, insert a one-character replacement for multiple Arabic characters that represent a single English character, and consider vowels as diphthongs.

Using a generated data set, we have found the NORM algorithm to be superior to all other algorithms, and REditext to be the second best, followed by LCS and Sgrams.

When using a manually transliterated data set, string similarity algorithms outperform the phonetic algorithms and our NORM algorithm. However, the REditex algorithm has been shown to be superior to all algorithms. LCS performed well in this data set, followed by Sgrams, Edit distance and AEditex.

We tested all algorithms in an IR experiment to investigate their effectiveness in capturing foreign words within a large collection of text. AEditex, NORM1, NORM2, NORM3, and NORM algorithms improved the recall of the light11 stemmer significantly, and improved MAP by over 7%. The improvement in MAP is weakly significant when using the AEditex, NORM1, and NORM algorithms.

String similarity algorithms performed well only for very high similarity thresholds (close to exact match). Phonetic algorithms and Gram Distance were the worst in this experiment, significantly decreasing the performance of the light11 stemmer.

We expanded foreign words in queries with their variants using the same algorithms to capture variants of words identified to be foreign in queries, both automatically and manually. Unranked lists of variants returned by phonetic algorithms were ordered using the Dice measure and then the top 3, 5, 10, 20, 30, 40, 50, or 100 words from the list of variants returned by each algorithm were used to replace their equivalent foreign word in the query. The best results were achieved by the normalisation and phonetic algorithms, with the best result recorded by the Soutex4 algorithm when expanding queries with the top 50 words returned as variants to foreign words in queries. The algorithm improved the light11 stemmer by 13.19% in the MAP measure, which is a statistically significant improvement at the 95% confidence level.

Our results show that normalising or expanding queries that have foreign words can enhance Arabic retrieval and that AIR systems must cater for common spelling variants; our results help understand how to find these in Arabic text.

Chapter 8

Conclusions and Future Work

In this thesis, we have investigated several techniques to improve Arabic text retrieval. We have improved light stemming by introducing rules that use the lexicon to distinguish core letters from actual prefixes and suffixes, tested the effectiveness of AIR systems on a large text collection, introduced algorithms that distinguish foreign words from native ones, and developed algorithms that conflate their variants in Arabic text. This chapter presents our conclusions, summarises our key contributions, and discusses possible directions for future work.

8.1 Improving Light Stemming Using Morphological Rules

In Chapter 4, we compared the performance of existing AIR systems and showed that the light10 stemmer is more effective than other stemmers. However, it is not as effective as the Buckwalter stemmer when using relevance feedback. We introduced new stemming techniques that minimised stemming mistakes in light stemming and led to improved retrieval results in some cases. We used the light10 stemmer as our underlying framework to evaluate the techniques that we developed. We extended word normalisation for improved retrieval effectiveness, and showed that automatic generation of stopword variants led to a reduction in precision and recall. We then introduced new techniques to remove the single-character prefixes: prepositions and conjunctions. We empirically showed that these techniques accurately remove prefixes, and as a result, aid retrieval effectiveness. Of the techniques we introduced — RPR, RR, RC, RCL, and RPRRC — RPRRC, in which we remove particles by duplicating the first character and removing the second character if it is a particle by checking the remaining string in the lexicon, performed the best.

Most morphological analysers use a list of pre-prepared stems, prefixes, suffixes, rules and patterns [Beesley, 1991; Khoja and Garside, 1999; Buckwalter, 2002]. Our affix-removal technique using a lexicon differs from previous techniques. It is concise and uses the different forms of Arabic words that exist in an Arabic lexicon to validate affixes before stemming. We use grammatical and morphological rules of Arabic words to validate affixes. Our approach is also different from light stemming in that it distinguishes core letters from actual affixes in Arabic words.

We showed that using a list of unique words found in an Arabic collection not only leads to better results, but also efficiently outperforms using professionally prepared lexicons. We presented novel techniques to remove different prefixes and suffixes, and showed that these techniques improve retrieval effectiveness.

Based on our observations on the effects of removing different prefixes and suffixes, we modified the light10 stemmer and developed three new versions: light11, light12 and light13. The three versions perform slightly better than the light10 stemmer, with light13 improving recall significantly when using relevance feedback.

In another experiment, we have tested the effectiveness of techniques used to improve Arabic text retrieval on a noisy data set. Using text automatically generated from a TV news soundtrack and machine-translated queries, we showed that using normalisation, stopping and light stemming improves retrieval effectiveness, but that n-grams and root stemming are not helpful.

Future Work

Despite the fact that morphology produces better correct stems than light stemming, stems are not always perfect in indexing Arabic words, as they are ambiguous without diacritics or considering context. Such ambiguity leads to conflation of similarly spelled words with different meanings under one indexing term. For example, consider the word "طالب" in the two sentences "طالب محمد بحقه" (/t̪ˤalaba muħmmad̪un biħaqihi/⟨Mohammed demanded his right⟩) and "محمد طالب ذكي" (/muħammad̪un t̪ˤalibun ðakijun/⟨Mohammed is a clever student⟩). While this word is a verb with the meaning "demand" in the first sentence, it is a noun with a different meaning "student" in the second. Such words, although spelt the same, should be indexed differently using two index terms. We plan to investigate techniques such as word disambiguation to distinguish such words while stemming Arabic.

8.2 The Effects of Large Text Collections on AIR

In Chapter 5, we investigated the effects of using a larger text collection. We built a new test collection of 90 topics with their respective relevance judgements using the AGW document collection. We used 20 assessors to propose topics and then identify relevant documents in the collection using the interactive searching and judging (ISJ) approach. This collection is far larger than those previously available for AIR, and our query set and ground truth judgments are valuable resources for future research. The topics and their relevance judgement are publicly available at http://goanna.cs.rmit.edu.au/~nwesri/Research/AGW/.

We used the new test collection to evaluate existing AIR approaches. Our results are consistent with those obtained using the TREC 2001 and TREC 2002 topics. The B.Stem, Al-StemN and light10 stemmers performed the best, while the Khoja root stemmer performed the poorest. Although the B.Stem and Al-StemN approaches perform slightly better than the light10 stemmer, the difference is not significant. When using relevance feedback, the B.Stem and light10 stemmers produce the highest MAP, while Al-StemN and B.Lemma produce the highest recall. We compared the performance of our approaches to the best existing AIR approaches (light10 and B.Stem), and showed that our approaches produce better precision and recall without relevance feedback. When using relevance feedback, our approaches showed slightly lower precision and recall than the light10 and B.Stem algorithms. We showed that our proposed approaches conflate terms in the corpus better than other algorithms, and that using the corpus as a background lexicon gives better results than using a professionally prepared lexicon.

Values for the parameters in the Okapi BM25 similarity function affect the effectiveness of IR systems by varying the impact of terms in document collections and queries. The optimal values for these parameters determined for English text collections have been used in AIR experiments [El-Khair, 2003; Darwish and Oard, 2003a; Darwish et al., 2005]. We have found that these values are not the best for the TREC 2001 Arabic collections. We have shown that when using the AGW Arabic collection, the best value for the b parameter is 0.25, the best value for the k_1 parameter is 1, and that changing k_3 has no effect on retrieval performance. With the new parameter values, performance increased significantly over the default values determined for English documents from the TREC 8 corpus. Similarly, we determined the parameter values that work best for the TREC 2001 and TREC 2002 collections which are not the same as those determined for the AGW collection, nor those determined for the TREC 8 English collection. Our findings show that the parameters that work better for

English collections do not necessarily work-well for Arabic collections. Our results indicate that these parameters differ across collections and should be determined for every individual collection, and that when using short queries, the b parameter has the most effect on retrieval performance.

Using the AGW collection, we showed that stemming improves effectiveness, but highlighted that the improvement is smaller than when using a smaller collection, such as the TREC 2001 collection (23.6% versus 100%). This is an important finding that indicates the need to improve stemming in Arabic. Experiments using this collection also indicate that root stemming is not a good option for indexing large collections of Arabic text documents.

Our conclusion based on our experiments using the TREC and AGW collections is that supporting light stemming with morphological rules aids retrieval effectiveness. This resulted in performance comparable to light stemming. We found that adding relevance feedback significantly improves the morphological rule results for the TREC collections, but that the corresponding results for the AGW collection are better without relevance feedback. Intensive morphological analysis — performed using the Buckwalter stemmer — aid retrieval effectiveness; however, the time required for this is unacceptably high compared to our approaches and light stemming.

Future Work

Our new test collection was created using the ISJ method. One of the main reasons behind using such a method is the lack of algorithms that capture different variants of foreign words. Since we have developed several such algorithms, we can now explore using pooling to identify documents to be judged. Another important direction to our research is developing a collection from an crawl of Arabic web documents, not constrained to news agency dispatches or news outlet web sites.

Arabic-language documents that are published on the Web differ both in style and in noisiness from the newswire dispatches used in most AIR research, and are likely to behave differently with many of the algorithms we have described in this thesis. Several issues we need to consider when building a web-based text collection include the different Arabic character encodings, the different styles of writing used by individuals, and detection of content in languages such as Persian and Urdu that share a same core alphabet with Arabic.

8.3 Identification of Foreign Words in Arabic Text

In Chapter 6, we showed that foreign words in Arabic text can be identified. We investigated the effectiveness of using lexicons, patterns, and n-grams for this purpose. We showed that the lexicon approach outperforms the other approaches, and described improvements to minimise false positives. These rules result in improved precision, but have a negative impact on recall. We showed that word frequency alone cannot be used to identify foreign words in Arabic text, but that it can be used to filter out most Arabic words prior to the foreign-word identification process. We improved the n-gram approach that uses language profiles generated from foreign words and Arabic native words. We determined that using the 1 700 most frequent n-grams from grams of size 1, 2, 3, 4, and 5 in each language is the best option. We also determined the best threshold for deciding whether a word is foreign. We combined the lexicon approach and the n-gram approach to improve identification, resulting in 80% recall and 93% precision for our target list of foreign words.

We determined that not stemming foreign words in Arabic text negatively affects retrieval effectiveness in both light stemming and root stemming. In contrast, removing the first letter if the remaining string exists within the list of foreign words results in improved performance, but not significantly.

Future Work

To improve identification of foreign words in Arabic, we plan to test several techniques. We plan to improve the n-gram technique by including not only the most frequent n-grams in language profiles, but also including the least frequent n-grams. We also plan to test the approach followed recently by Goldberg and Elhadad [2008] to identify foreign words in Hebrew. In this approach, we plan to use a pure native Arabic text collection and a list of transliterated words to train a statistical model to learn the pattern of foreign words in Arabic text.

8.4 Conflation of Foreign Word Variants in Arabic Text

Foreign words in Arabic are characterised by multiple spellings. Conflating such words is not possible using stemming as they have different morphological structure than Arabic native words. In Chapter 7 we investigated conflating the different versions of transliterated foreign words in Arabic text. We developed different versions of the NORM algorithm to normalise

foreign words in Arabic. We started by removing vowels from foreign words (NORM1), then kept the first and the last characters of the word (NORM2), replaced multiple Arabic characters that correspond to a single English character with a single normalised equivalent (NORM3), and considered vowels and diphthongs (NORM). We developed the Soutex algorithm, a Soundex-like algorithm specifically developed to collapse variants of foreign words in Arabic, extended the English Editex algorithm to Arabic in the AEditex algorithm, and further enhanced this to produce better ranking in the REditex algorithm. We compared the performance of these algorithms with major alternatives developed for English and Arabic: gram count (gramCount), gram distance (gramDist), Dice, edit distance (Edit Distance), longest common subsequence (LCS), and skip grams (Sgrams), and Asoundex-Final.

Using a generated data set, we found the NORM algorithm to produce the best average precision (66%), followed by REditex (65%), LCS (61%), and Sgrams (59%). When using a manually transliterated data set, string similarity algorithms outperformed the phonetic algorithms and our NORM algorithm. However, the REditex algorithm was superior to all other algorithms, achieving an average precision of 82%. LCS was the second best (78%), followed by Sgrams (76%), Edit Distance (70%)and AEditex (62%).

We tested all algorithms in an IR context to investigate their effectiveness in supporting AIR systems in finding documents relevant to queries containing transliterated foreign words. We found that the AEditex, NORM1, NORM2, NORM3, and NORM algorithms improved the recall of the light11 significantly, contributed a weakly significant improvement in MAP, and improved P@10 and R-Precision. These algorithms increased MAP by more than 8%.

We used the same algorithm to expand foreign words in Arabic queries with their variants. Unranked lists of variants returned by phonetic algorithms were ordered using the Dice measure. We selected in turn the top 3, 5, 10, 20, 30, 40, 50, and 100 words from the list of variants returned by each algorithm to use alongside the foreign word in the query. The best results were achieved by the normalisation and phonetic algorithms, with the best result recorded by the Soutex4 algorithm when expanding queries with the top 50 words. The Soutex4 algorithm improved the light11 stemmer by 13.19% in the MAP measure, which is a statistically significant improvement at the 95% confidence level.

Future Work

There are several additional algorithms that could be used to find variants of foreign words in Arabic. These include the Damerau-Levenshtein Distance [Damerau, 1964], which is similar

to Edit Distance but considers a transposition operation as equal to a deletion, insertion, or substitution operation with a cost of 1; and the Jaro and Winkler similarity measures [Winkler, 1990] that compute the similarity between two strings by comparing the common characters in the first half of the two strings and considering the number of transpositions. Another direction that we intend to investigate is considering the words surrounding a possible foreign word. In general, foreign names appear in full, with first and last names appearing together when they first are mentioned in text, but the last name is used often by itself within the text. For example, "بيل كلينتون" (/bjlkljnṯwn/⟨Bill Clinton⟩) and "كلينتون" (/kljnṯwn/⟨Clinton⟩) are used interchangeably to represent the same person. Techniques that identify person names such as named entity recognition can be utilised to normalise names correctly. Moreover, in many instances transliterated words are joined together, while appearing as two independent words in others. For example, the name "Condoleeza Rice" is sometimes found as one word "كوندوليسارايس" /kwnḍwljsarajs/ as well as two separate words as "كوندوليسا" /kwnḍwljsa/ and "رايس" /rajs/. We plan to deal with such cases following the same approach we used to deal with Arabic compound nouns.

8.5 Concluding Remarks

We have presented the first in-depth empirical comparison of stemming, indexing, and foreign word identification and normalisation for Arabic using a range of collections, including a new collection that is much larger than those used previously in this domain. We believe that this thesis contributes greatly to the understanding of IR for a language spoken by people in more than 23 countries, and familiar to over 1 billion people around the world.

Appendix A

AGW Topics

In this Appendix, we show the AGW Arabic topics used in our experiments in Chapters 5, 6, and 7. Table A.1 on page 239 shows the number of relevant and non-relevant documents for each query (topic).

<top>
<numb> Number: 1
<title>الزلازل التي وقعت في المغرب
<desc> Description:
الزلازل التي ضربت المغرب
<narr> Narrative:
المقالات التي تتحدث عن الزلازل التي ضربت المغرب والتي
تسببت في تضرر ١٠٠٠ لاجئ. كذلك التي خلفت ١٠٠ قتيل في جنوب المغرب.
</top>

<top>
<numb> Number: 2
<title>حلف شمال الأطلسي الناتو
<desc> Description:
عدد الدول المنضمة اخيرا للمنظمة
<narr> Narrative:
المقالات التي تتحدث عن إنضمام ٥ دول من شرق اوروبا منهم بلغاريا
ولتوانيا إلى حلف الناتو، كذلك المقالات التي تتحدث عن إرسال ٥٠٠٠
جندي من حلف الناتو الى البوسنة و مقتل ١٠٠ عسكري من الناتو في عملية إغتيال.
</top>

<top>
<numb> Number: 3
<title>الحصار على ليبيا
<desc> Description:
رفع الحصار على ليبيا
<narr> Narrative:
المقالات التي تتحدث عن قرار الامم المتحدة برفع الحصار على ليبيا، كذلك
قرار مجلس الامن بفرض حصار اقتصادي على ليبيا ورفض ليبيا لهذا القرار.
كذلك الخسائر التي تكبدتها ليبيا من جراء الحصار.
</top>

<top>
<numb> Number: 4
<title>أحداث مدرسة بيسلان الروسية
<desc> Description:
من وراء أحداث مرسة بيسلان؟
<narr> Narrative:
كافة المقالات التي تتحدث عن الجماعة التي احتجزت ١٠٠٠ طالبا في مدرسة بيسلان وكيف قامت السلطات الروسية باخلاء الرهائن. وما هي الاجراءات الامنتية التي اتخذتها هذه المدرسة بعد احدات الاعتقال.
</top>

<top>
<numb> Number: 5
<title>أسلحة الدمار الشامل العراقية
<desc> Description:
ما مدى خطورة أسلحة الدمار الشامل العراقية؟
<narr> Narrative:
كافة المقالات التي تتحدث عن أسلحة الدمار الشامل العراقية وتقرير السي آى ايه بهذا الخصوص. كذلك المقالات التي تتحدث عن رفض الادارة العراقية لدخول مفتشي الامم المتحدة للمواقع العراقية، والتي تتحدث ايضا عن زيارة وفد من منظمة الطاقة الدولية إلى العراق.
</top>

<top>
<numb> Number: 6
<title>مخزون البترول في العراق
<desc> Description:
كم يبلغ مخزون العراق من البترول وكم هي نسبته من المخزون العالمي؟
<narr> Narrative:
اي اخبار او مقالات تتحدث عن مخزون العراق من النفط ونسبته من المخزون العالمي. المقالات التي تتحدث عن مخزون الولايات المتحدة الامريكية وغيرها من الدول ليس لها علاقة.
</top>

<top>
<numb> Number: 7
<title>مرض جنون البقر
<desc> Description:
ما هي اسباب مرض جنون البقر؟
<narr> Narrative:
كافة المقالات التي تتحدث عن اسباب مرض جنون البقر والاضرار الاقتصادية التي سببها. المقالات التي تتحدث عن مرض جنون البقر بصورة عامة او اكتشافه في الدول المختلفة ليس له علاقة بالموضوع.
</top>

<top>
<numb> Number: 8
<title>الحاكم المدني بول برمير
<desc> Description:
من هو وما هو تاريخه؟
<narr> Narrative:
كافة المقالات التي تتحدث عن شخصية الحاكم المدني بول برمير وتاريخه. ما هو عمره وما هي وظيفته السابقة قبل ان يعين كحاكم مدني للعراق. المقالات التي تتحدث عن تصريحات الحاكم بول برمير في العراق ليس لها علاقة بالموضوع. كذلك كافة المقالات التي تتحدث عن الاجراءات الادارية التي قام بها في العراق ليس لها علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 9
<title>دور كولن باول في حرب العراق
<desc> Description:
من هو كولن باول وما هو دوره في الحرب الامريكية العراقية؟
<narr> Narrative:
كافة المقالات التي تتحدث دور كولن باول في الحرب الامريكية العراقية. المقالات التي تتحدث عن شخصية كولن باول ليس لها علاقة. كذلك المقالات التي تتحدث عن دوره في السياسة الامريكية ليس لها علاقة.
</top>

<top>
<numb> Number: 10
<title>اعتقال خالد شيخ محمد ومصطفى احمد هوساوي في الباكستان
<desc> Description:
كيف تم اعتقال كل من خالد شيخ محمد ومصطفى احمد هوساوي في الباكستان
<narr> Narrative:
الوثائق المطلوبة هي التي تتحدث عن كيفية القبض عليهما. الوثائق التي
تتكلم عن القاعدة وحركة طالبان غير مطلوبة.
</top>

<top>
<numb> Number: 11
<title>اعتقال رضوان عصام الدين حنبلي في بانكوك ـ تايلاند
<desc> Description:
ما هي الجماعة الاسلامية في تايلاند ومن هو رضوان عصام الدين الملقب بحنبلي؟
<narr> Narrative:
المقالات التي تعطي فكرة عامة عن الجماعة الإسلامية في تايلاند وقادتها.
المقالات التي تتحدث عن الجماعات الاسلامية في الدول الأخري غير مطلوبة.
</top>

<top>
<numb> Number: 12
<title>أخر رحلة لطائرة الكونكورد
<desc> Description:
معرفة اسباب وقف استعمال الكونكورد في الرحلات الجوية.
<narr> Narrative:
الوثائق التي تتحدث عن عيوب هذا النوع من الطائرات المدنية واسباب وقف العمل بها.
الوثائق التي تتحدث عن كيفية صناعة الكونكورد غير مرغوب بها.
</top>

<top>
<numb> Number: 13
<title>اعتقال صدام حسين
<desc> Description:
كيف تم اعتقال الرئيس العراقي السابق صدام حسين؟
<narr> Narrative:
الوثائق المطلوبة هي تلك التي تتحدث عن كيفية اعتقال صدام حسين. الوثائق
التي تتحدث عن ممارسات صدام حسين خلال فترة حكمه ليس لها علاقة بالموضوع.
</top>

<top>
<numb> Number: 14
<title>زلزال مدينة بام في إيران
<desc> Description:
ما مدي الدمار الناتج عن زلزال مدينة بام في ايران؟
<narr> Narrative:
الوثائق التي تتحدث عن الدمار الناتج عن هذا الزلزال هي المطلوبة.
اما الوثائق التي توضح مدي تعاون المجتمع الدولي مع إيران في هذه الازمة غير مطلوبة.
</top>

<top>
<numb> Number: 15
<title>إنفجار مركبة الفضاء كولومبيا فوق مدينة تكساس
<desc> Description:
ما هي اسباب انفجار مركبة الفضاء كولومبيا؟
<narr> Narrative:
الوثائق المطلوبة تلك التي تتحدث عن الأسباب التي ادت الى انفجار
هذا المكوك. اما الوثائق التي تتحدث عن كلفة صناعة هذا المكوك وعن
تفاصيل الرحلة غير مطلوبة.
</top>

<top>
<numb> Number: 16
<title>اتهام محكمة تابعة للامم المتحدة لـ ٢١ قائد حربي لجرائم ضد الانسانسية
<desc> Description:
من هم هؤلاء القادة وهل تم فعلا قضائهم والقبض عليهم؟
<narr> Narrative:
الوثائق التي تحتوي على اسماء هؤلاء القادة ومواعيد محاكمتهم هي المطلوبة. الاحداث والمقالات التي تحدث عن الحرب في البوسنة والتي تتحدث عن محاكمات اخري بجرائم ضد الانسانية وكذلك المقالات التي تتحدث عن الامم المتحدة ليس لها علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 17
<title>علاقة رئيس الوزراء الايطالي جيوليو اندريوتي بالمافيا
<desc> Description:
ما هي نوع علافة رئيس الوزراء الايطالي جيوليو اندريوتي بعصابة المافيا؟
<narr> Narrative:
التقارير التي تحتوي على مستندات وصور تبرهن وجود علاقة بين رئيس الوزراء الايطالي اندريوتي بالمافيا والتقارير التي تؤكد دور المافيا في وصول اندريوتي إلى رئاسة الحكم في الحكومة الايطالية هي المطلوبة. التقارير التي تتحدث عن رئيس الوزراء الايطالي جيوليو لوحده او المافيا فقط ليس لها علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 18
<title>ضحايا موجات الحر بالغرب الاوسط للولايات المتحدة سنة ١٩٩٥
<desc> Description:
ما هو عدد ضحايا موجات الحر بالغرب الاوسط للولايات المتحدة الامريكية في ٢٩/٦/١٩٩٥ وكم بلغت درجات الحرارة فيها؟
<narr> Narrative:
المقالات التي لها علاقة بهذا الموضوع هي التي تبين العدد الاجمالي للضحايا مع الاشارة الى اكثر المدن تضررا بسبب الحرارة. المقالات الاخبارية التي تبين تأثير التغيرات المناخية بالعالم مع امثلة مباشرة لبعض المناطق المتأثرة ليس لهاعلاقة بموضوع البحث.
</top>

<top>
<numb> Number: 19
<title>رائد الفضاء الروسي فولري بولياكوف
<desc> Description:
ما هي المدة التي قضاها الرائد الروسي فولري بولياكوف في الفضاء الخارجي؟
<narr> Narrative:
المقالات التي تبين تاريخ المغادرة وتاريخ العودة لرائد الفضاء الروسي فولري بولياكوف مع طاقمه والمدة التي بقاها في الفضاء. المصاعب التي واحهت الرائد الروسي. ما ليس له علاقة هو المقالات التي تبين الدور الروسي في اكتشاف لفضاء والتي تتحدث عن شخصيات اخري تحمل نفس اللقب لبولياكوف
</top>

<top>
<numb> Number: 20
<title>جاك شيراك
<desc> Description:
متى تم انتخاب جاك شيراك كرئيس لفرنسا؟
<narr> Narrative:
المقالات التي تحتوي على تاريخ انتخاب الرئيس الفرنسي جاك شيراك كرئيس لدولة فرنسا، ومعلومات عن الرئيس الفرنسي جاك شيراك. ما ليس له علاقة بالموضوع هو معلومات عن الانتخابات في فرنسا ودور جاك شيراك في الاحداث العالمية كحرب الخليج.
</top>

<top>
<numb> Number: 21
<title>الإسلام في فرنسا
<desc> Description:
الإجراءات والقوانين التي أتخذتها الحكومة الفرنسية إزاء منع ارتداء الحجاب في فرنسا
<narr> Narrative:
الأسباب التي جعلت الحكومة الفرنسية تتخذ هذه الخطوة. ومارأي الرئيس الفرنسي جاك شيراك في هذا الأمر. وماهي ردود الفعل في العالم الاسلامي.
</top>

<top>
<numb> Number: 22
<title>بيل كلينتون ومونيكى
<desc> Description:
الأحداث التي جرت بين مونيكى لوينسكى والرئيس الأمريكي بيل كلينتون
<narr> Narrative:
تفاصيل القضية ورد فعل الكنقورس الأمريكي. وهل هي قضية مفتعلة من الحزب الجمهوري؟
</top>

<top>
<numb> Number: 23
<title>المقرحي ولوكربي
<desc> Description:
القضية التي اتهمت بها ليبيا بتفجير طائرة البنام الأمريكية وفوق قرية لوكربي الأسكتلندية.
<narr> Narrative:
كيفية حصولها وطريقة تعامل ليبيا مع هذه القضية، وهل هي قضية سياسية ام جنائية؟
</top>

<top>
<numb> Number: 24
<title>استنساخ الحيوانات
<desc> Description:
كيف واين تم استنساخ النعجة دوللي ؟
<narr> Narrative:
المقالات المتوقعة هي التي تتحدث عن استنساخ النعجة دوللي اول حيوان مستنسخ
من خلية جسمية والحيوانات عموما. المقالات التي ليس لها علاقة هي تلك التي
تحتوي على الشيخوخة المبكرة في الحيوانات المستنسخة كما هو الحال بخصوص النعجة دوللي.
</top>

<top>
<numb> Number: 25
<title>الثائرين في الصين
<desc> Description:
من هم الثوار الصينينون ومتى كانت هذه التورة.
<narr> Narrative:
كافة المقالات التي تتحدث عن الثورة في الصين ومتي حدثت.
</top>

<top>
<numb> Number: 26
<title>كوارث الحج
<desc> Description:
ما مدى استعداد السعودية لتفادي كوارث الحج ؟
<narr> Narrative:
المقالات المتوقعة هي تلك التي تتحدث عن كوارث الحج بالأرقام والتواريخ وكيف تستعد السعودية لتفادي كوارث الحج. المقالات الغير متوقعة هي تلك التي تتحدث عن الحج بصورة عامة.
</top>

<top>
<numb> Number: 27
<title>مجزرة حوش خميستي
<desc> Description:
متى وقعت مجزرة حوش خميستي؟
<narr> Narrative:
المقالات المتوقعة : تاريخ وقوع المجزرة عدد القتلى في المجزرة المقالات الغير متوقعة: من وراء مجازر الجزائر
</top>

<top>
<numb> Number: 28
<title>فرنسا وزائير
<desc> Description:
الدور الفرنسي في زائير
<narr> Narrative:
المقالات المتوقعة : دعم فرنسا لحكومة زائير الجديدة : دور فرنسا في الاقتصاد الزائيري : مدى الدعم العسكري الفرنسي الحكومية. المقالات الغير متوقعة : النجم الرياضي الفرنسي المولود في زائير
</top>

<top>
<numb> Number: 29
<title>سيراليون والانقلابات العسكرية
<desc> Description:
الحركات الانقلابية في سيراليون
<narr> Narrative:
المقالات المتوقعة : الثورة في سيراليون وتعدد الانقلابات، : متمردي سيراليون والانقلابات المقالات الغير متوقعة :سيراليون نحو تعزيز السلام عقب عشر سنوات تقريبا من الحرب الأهلية الوحشية و الإنقلابات.
</top>

<top>
<numb> Number: 30
<title>تفجيرات الحادي عشر من سبيتمبر في أمريكا
<desc> Description:
عدد الخسائر البشرية في تفجيرات الحادي عشر من سبتمبر في امبركا.
<narr> Narrative:
كافة المقالات التي تحتوي على عدد ضحايا هذه التفجيرات وعدد الجرحى . الوثائق التي تحتوي على الخسائر المادية بكل انواعها ليس لها صلة بالموضوع.
</top>

<top>
<numb> Number: 31
<title>حقن أطفال ليبيا بفيروس الايدز
<desc> Description:
عدد الأطفال المحقونين بفيروس الايدز في ليبيا
<narr> Narrative:
المقالات المتوقعة هي التي تحتوي على عدد الاطفال المصابين بفيروس الايدز.
اما المسؤولين على حقن الفيروس او آلية نقل الفيروس لهم ليست مهمة هنا.
</top>

<top>
<numb> Number: 32
<title>قضية لوكربي وحكم المحكمة العليا في جلاسكو
<desc> Description:
ما حكم المحكمة العليا في جلاسكو بشأن قضية لوكربي
<narr> Narrative:
المقالات التي لها علاقة بموضوع البحث هي تلك التي تحتوي على نص الحكم
على الليبي في قضية لوكربي فقط.
</top>

<top>
<numb> Number: 33
<title>اعتقال الموظف روبرت هانسن
<desc> Description:
البراهين التي ادت الى اعتقال الموظف روبرت هانسن في قضية التجسس
<narr> Narrative:
كافة المقالات التي تحتوي علي الدلائل التي قبض على اثرها الموظف
روبرت هانسن. كذلك التي تحتوي على مجريات القضية.
ما يتحتوي على قضايا التجسس والحرب الباردة عموما ليس له علاقة بالموضوع.
</top>

<top>
<numb> Number: 34
<title>محاكمة سلوبودان ميلوسيفتش
<desc> Description:
عدد التهم التي وجهت الى سلوبودان ميلوسيفتش اثناء المحاكمة
<narr> Narrative:
سرد وتفصيل للتهم الموجهة اليه، تغطية اعلامية للقضية، التاريخ
السياسي ل سلوبودان ميلوسيفتش.
</top>

<top>
<numb> Number: 35
<title>قتل الأمير ديبندرا لوالده الملك
<desc> Description:
الأسباب التي دفعت الأمير ديبندرا لقتل والده الملك
<narr> Narrative:
المقالات المطلوبة هي التي تناولت بعض الأسبات المتوقعة لقتل الملك،
كذلك التي تحتوي على سيرة العائلة المالكة النيبالية و أحداث قتل الملك.
</top>

<top>
<numb> Number: 36
<title>الهجوم على مطار باندارانايك
<desc> Description:
ما هي حجم الخسائر البشرية إثر الهجوم على مطار باندارانايك
<narr> Narrative:
الخسائر المادية والبشرية للهجوم، أسباب الهجوم، تغطية اعلامية لعملية الهجوم.
</top>

<top>
<numb> Number: 37
<title>تحطم الطائرة الروسية المتوجهة الي نوفوسيبيرسك
<desc> Description:
هل تم تعويض اهالي ضحايا الطائرة الروسية المتوجهة الى نوفوسيبيرسك
<narr> Narrative:
تحطم الطائرة، عدد الضحايا، عملية انتشال الجثث، حجم التعويضات
التي دفعت من قبل الشركة الروسية
</top>

<top>
<numb> Number: 38
<title>انتخاب الرئيس أليجاندرو توليدو
<desc> Description:
عدد الأصوات التي تحصل عليها الرئيس أليجاندرو توليدو في الانتخابات
<narr> Narrative:
تغطية اعلامية للحملة الانتخابية، حجم الأموال التي صرفت في الحملة،
صور للرئيس، نسبة الأصوات التي تحصل عليها المنتخبين.
</top>

<top>
<numb> Number: 39
<title>تسونامي
<desc> Description:
زلزال تسونامي يضرب دول جنوب شرق اسيا
<narr> Narrative:
اضرار زلزال تسونامي على جنوب شرق آسيا، اسباب حدوث الزلزال ،
مخلفات الزلزال ، اعداد الضحايا الزلازال، الدول المعرضة للزلزال.
</top>

<top>
<numb> Number: 40
<title>سبيريت وابورتيونيتي واكتشاف الماء على سطح المريخ
<desc> Description:
كيف كان رد فعل ناسا على هذا الاكتشاف؟
<narr> Narrative:
تقارير ناسا، نتائج هذه الرحلة، مصاعب هذه الرحلة ، تنافس عديد
من الشركات حول دخول ميدان السياحة الفضائية.
</top>

<top>
<numb> Number: 41
<title>دخول اريل شارون للحرم الشريف بالقدس
<desc> Description:
لماذا قام اريل شارون بزيارة الحرم الشريف بالقدس؟
<narr> Narrative:
كافة المقالات التي تتحدث عن الزيارة المتعمدة لاريل شارون للحرم
الشريف بالمسجد الاقصي بالقدس. ما هي نتائج هذه الزيارة وما هو الهدف منها.
المقالات التي تتحدث عن الحرم الشريف وموقف اريل شارون منه ليس لها علاقة بالاستفسار.
</top>

<top>
<numb> Number: 42
<title>مسجد فرهاديا في بانجا لوكا في البوسنة والهرسك
<desc> Description:
كيف تم تهديم مسجد فرهاديا في البوسنة الهرسك وكيف تم اعادة بناؤه؟
<narr> Narrative:
كافة المقالات التي تتحدث عن هذا المسجد. اين يقع وكيف تم تهديمه وكيف
تم محاولة بناؤه وما هي النتائج التي سببها محاولة اعادة بنائه وهل تم اعادة بنائه ام لا.
</top>

<top>
<numb> Number: 43
<title>حصار كنيسة المهد
<desc> Description:
احدات الحصار الاسرائيلي لكنيسة المهد بعد احتماء عدد من الفلسطينين بداخلها.
<narr> Narrative:
كافة المقالات التي تتحدث عن لجوء الفلسطينين اليها وكيف تم
الحصار وكيف تم فكه. المقالات التي تتحدث عن الكنيسة او الممارسات
الصهيونية ضد الفلسطينين ليس لها علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 44
<title>إعصار سنلاكو بالصين
<desc> Description:
اين ومتى يحدث اعصار سنلاكو بالصين الشعبية؟
<narr> Narrative:
كافة المقالات التي تتحدث عن هذا الاعصار. المقالات التي تتحدث عن
الاعاصير الأخرى ليس لها علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 45
<title>موجات الحرارة في اوروبا
<desc> Description:
الموجات الحرارية التي مرت باوروبا.
<narr> Narrative:
كافة المقالات التي تتحدث عن موجات الحرارة التي ضربت الدول الاوربية
والاضرار التي خلفتها. الموجات الحرارية في دول العالم الاخر ليس لها
علاقة بموضوع البحث كذلك النينو وظاهرة الاحتباس الحراري.
</top>

<top>
<numb> Number: 46
<title>المجرم هارولد شيبمان
<desc> Description:
من هو المجرم هارولد شيبمان وما هي الجرائم التي ارتكبها.
<narr> Narrative:
كافة المقالات التي تتحدث عن الجرائم التي ارتكبها المجرم هارولد
شيبمان وملابسات قضية موته في زنزانته.
</top>

<top>
<numb> Number: 47
<title>مباريات ارسنال ومانشستر يونايتد
<desc> Description:
متي وكم كانت نتيجبة لقاءات كل من ارسنال ومانشستر يونايتد
<narr> Narrative:
كافة المقالات التي تتحدث عن المبارايات بين هذين الفريقين فقط مع
النتائج الخاصة بكل مباراة. ما يتعلق باحد الاندية
دون الاخر ليس له علاقة بالموضوع.
</top>

<top>
<numb> Number: 48
<title>الدول التي تمكنت من الوصول الى الفضاء
<desc> Description:
ما هي الدول التي تمكنت من اطلاق صواريخ الى الفضاء الخارجي؟
<narr> Narrative:
كافة المقالات التي تتحدث عن اسماء الدول التي اطلقت صواريخ الى
الفضاء عدا امريكيا وروسيا. رجوع الصواريخ او الرحلات
المكوكية الامريكية والروسية ليست مهمة.
</top>

<top>
<numb> Number: 49
<title>معركة الفلوجة في العراق
<desc> Description:
ما هي احداث معركة الفلوجة بين المجاهدين والامريكين؟
<narr> Narrative:
كافة المقالات التي تتحدث عن احداث معركة الفلوجة بين المجاهدين
والامريكيين. كيف تم القضاء على المجاهدين وكيف تم الاستيلاء
على المدينة من قبل الامريكيين وما هي الخسائر الامريكية في هذه المعركة.
</top>

<top>
<numb> Number: 50
<title>مذبحة سربرينشيا
<desc> Description:
اين ومتى وقعت مذبحة سربرينشيا في البوسنة والهرسك ومن المسؤول؟
<narr> Narrative:
احداث هذه المذبحة وعدد القتلي واسبابها ومن المسؤول عنها.
</top>

<top>
<numb> Number: 51
<title>مجزرة دونبلان
<desc> Description:
لماذا قام توماس هاملتون بقتل ١٦ طالبا في مدرسة دونبلان؟
<narr> Narrative:
كافة المقالات التي تتكلم عن الحادثة والمقالات التي تتحدث عن
الاسباب التي جعلت توماس هاملتون يقوم بقتل ١٦ طالبا في
مدرسة دونبلان في اسكتلندا
</top>

<top>
<numb> Number: 52
<title>مجزرة ميناء ارثر
<desc> Description:
لماذا قالم مارتين برايانت بقتل ٣٥ في ميناء ارثر بتزمانيا؟
<narr> Narrative:
المقالات التي تتحدث عن ملابسات الحادثة واسبابها كذلك مصير مارتين برايانت.
والتي تتحدث عن الاسباب التي جعلت هذا الشخص يقوم بهذه المجزرة.
</top>

<top>
<numb> Number: 53
<title>تقاعد النجم مايكل جوردن
<desc> Description:
متى تقاعد النجم مايكل جوردن، وما هي الظروف التي على اثرها تقاعد اللاعب مايكل جوردن
<narr> Narrative:
المقالات التي تتعلق بتقاعد أو اعتزال النجم مايكل جوردن وما هي الاسباب التي ادت الى ذلك.
</top>

<top>
<numb> Number: 54
<title>وفاة الملك حسين ملك المملكة الاردنية
<desc> Description:
كيف توفى الملك حسين ملك الأردن
<narr> Narrative:
المقالات المتعلقة بوفاة الملك حسين ملك الاردن والمتعلقة بأسباب الوفاة.
</top>

<top>
<numb> Number: 55
<title>قضية الرحلة رقم ١٠٣ لشركة بانام ١٩٨٦
<desc> Description:
مت تم تسليم المشتبه الليبي في قضية بانام للسلطات الاسكتلندية.
<narr> Narrative:
المقالات المتعلقة بتسليم المشتبه به الليبي في قضية بانام للسلطات الاسكتلندية.
</top>

<top>
<numb> Number: 56
<title>حزب العمل يفوز بمكتب في الانتخابات العامة
<desc> Description:
حزب العمل في نيوزيلندا بزعامة هيلين يفوز بمكتب في الانتخابات العامة
<narr> Narrative:
كافة المقالات المتعلقة بفوز حزب العمل النيوزيلاندي بزعامة هيلين
بمكتب الانتخابات العامة. المقالات التي تتحدث عن السياسة الخارجية لهيلين ليست مطلوبة.
</top>

<top>
<numb> Number: 57
<title>السيطرة على قناة بنما
<desc> Description:
متى وكيف تم تحويل السيطرة على قناة بنما للبنميين
<narr> Narrative:
كافة المقالات التي تتحدث عن كيفية تحويل السيطرة على قناة بنما
الى البنميين. المقالات التي تتحدث عن استخدام قناة بنما فقط ليس لها علاقة.
</top>

<top>
<numb> Number: 58
<title>إحتفالات الألفية الثالثة في كرين وتش في لندن
<desc> Description:
متى تم الاحتفال بالألفية الثالثة بإفتتاح الملكة إليزبت الثانية للقبة الألفية في كرين وتش في لندن
<narr> Narrative:
كافة المقالات المتعلقة بالاحتفال بالألفية الثالثة والمقالات المتعلقة
بإفتتاح الملكة إليزبت الثانية للقبة الألفية في كرين وتش في لندن.
</top>

<top>
<numb> Number: 59
<title>بداية العمل بالعملة الاوروبية اليورو
<desc> Description:
متى بدا العمل بتداول العملة الأوروبية اليورو
<narr> Narrative:
كافة المقالات المتعلقة ببداية العمل بالعملة الأوربية الجديدة اليورو.
القالات التي تحتوي على اسعار العلملات بما في ذلك اليورو ليس لها علاقة.
</top>

<top>
<numb> Number: 60
<title>فيروس الايدز
<desc> Description:
كيف ينتقل فيروس الايدز
<narr> Narrative:
طريقة انتقال الفيروس (الدم ـ الحقن) معلومات عن تركيب الفيروس،
طريقة تشخيصه، طريقة الوقاية منه، انتشاره في العالم.
</top>

<top>
<numb> Number: 61
<title>التعداد السكاني في استراليا
<desc> Description:
كم عدد السكان في استراليا؟
<narr> Narrative:
احصائيات متعددة خلال عدة سنوات مختلفة مثلا من سنة ٢٠٠١ الى ٢٠٠٦.
سبب زيادة عدد السكان. الاحصائيات السكانية للدول الاخري ليس له علاقة.
</top>

<top>
<numb> Number: 62
<title>القواعد الامريكية في السعودية
<desc> Description:
سبب وضع القواعد الامريكية في السعودية
<narr> Narrative:
الوثائق التي تتحدث عن اسباب وضع القواعد الامريكية في السعودية
مثل حرب العراق مع امريكيا وحرب العراق مع الكويت و حرب الخليج
الاولى والثانية وطلب الحكام العرب العون من امريكيا.
</top>

<top>
<numb> Number: 63
<title>مجزرة قانا
<desc> Description:
كم عدد القتلي في قانا
<narr> Narrative:
معلومات عامة عن مكان وتاريخ هذه المجزرة، عدد القتلى، و من ارتكب هذه الجريمة.
</top>

<top>
<numb> Number: 64
<title>يونسكوم
<desc> Description:
سبب نزع السلاح العراقي
<narr> Narrative:
الحصار على العراق، الغذاء مقابل النفط، اتهام العراق بتملكها السلاح النووي، حرب الخليج الاولى
</top>

<top>
<numb> Number: 65
<title>إنتخاب جاك شيراك
<desc> Description:
متى تم انتخاب جاك شيراك رئيسا لجمهورية فرنسا
<narr> Narrative:
كافة المقالات المتعلقة بانتخاب الرئيس جاك شيراك رئيسا لجمهورية فرنسا
</top>

<top>
<numb> Number: 66
<title>فوز رئيس الوزراء جون ميجور
<desc> Description:
متى فاز رئيس الوزراء جون ميجور بمعركته ليبقى زعيم حزب المحافظين.
<narr> Narrative:
كافة المقالات المتعلقة بفوز رئيس الوزراء جون ميجور ليبقى رئيس حزب المحافظين
</top>

<top>
<numb> Number: 67
<title>الإفراج عن ديفيد دالبيرت و ويليم بارلون
<desc> Description:
متى تم الافراج عن ديفيد دالبيرت و ويليم بارلون في العراق
<narr> Narrative:
المقالات المتعلقة بإفراج الرئيس العراقي صدام حسين عن الجاسوسين
ديفيد دالبيرت و ويليم بارلون. الوثائق التي تتحدث عن كيفية اسر
هذين الجاسوسين او تمكن زوجيتيهما من زيارتهما في السجن وكذلك
الوثائق التي تحتوي على تصريحات الحكومة الامريكية بخصوصهما
ليس لها علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 68
<title>دخول النمسا وفنلندا والسويد للاتحاد الأوربي
<desc> Description:
متى تم دخول النمسا وفنلندا والسويد للإتحاد الأوروبي
<narr> Narrative:
المقالات المتعلقة بدخول هذه الدول في الاتحاد الاوروبي.
</top>

<top>
<numb> Number: 69
<title>إجراء تجارب نووية في الهند
<desc> Description:
ما هي التجارب النووية التي قامت بها الهند؟
<narr> Narrative:
نتائج التجارب النووية التي قامت بها الهند، ردود الفعل الدولية
إزاء هذه التجارب، التجهيزات التي قامت بها الهندلإقامة هذه التجارب،
الضجة الاعلامية الرافضة لإقامة هذه التجارب.
</top>

<top>
<numb> Number: 70
<title>حكم بالسجن المؤبد على تيري نيكولس في تفجير اوكلاهوما
<desc> Description:
ماالحكم الذي أصدر على تيري نيكولس لضلوعه في تفجير مدينة اوكلاهوما؟
<narr> Narrative:
نص الحكم على تيري نيكولس، الخسائر الناتجة عن التفجير، الاحداث التي وقعت عقب التفجير
</top>

<top>
<numb> Number: 71
<title>إنتخاب جون هورد
<desc> Description:
كيف تم انتخاب جون هورد كرئيس لاستراليا؟
<narr> Narrative:
إعادة انتخاب جون هورد لحكومة ائتلافية جديدة. الحملات الدعائية
لانتخاب جون هورد. المقالات التي تتحدث عن زيارات جون هورد للدول الاخرى
وتصريحاته بشان العراق او السياسة الخارجية ليس لها علاقة.
</top>

<top>
<numb> Number: 72
<title>تأثيرات إعصار ميتش
<desc> Description:
ما هي الاضرار الناشئة عن اصابة اعصار ميتش لأمريكا الوسطى؟
<narr> Narrative:
التأثيرات الناجمة عن الاعصار، خريطة مرور الاعصار، تأثير الأعاصير في امريكا الوسطى.
</top>

<top>
<numb> Number: 73
<title>إطلاق مكوك فضائي ياباني إلى المريخ
<desc> Description:
ماالغرض وراء إطلاق المكوك الفضائي الياباني إلى المريخ ؟
<narr> Narrative:
إستكشاف الفضاء الخارجي بمكوك ياباني. مسارات القضاء الخارجي، وصف
المكوك الفضائي المعد لهذا الغرض، وصف واحداث الرحلة الى المريخ.
</top>

<top>
<numb> Number: 74
<title>بلاي ستايشن
<desc> Description:
المواصفات ـ الصيانة ـ التعليمات ـ الاسعار. المبيعات في دول العالم ـ الألعاب المنسوخة
<narr> Narrative:
الجيل الثاني من منصة الالعاب السوني ـ الاسعار ـ الاسواق ـ
المواصفات ـ الوكلات في الدول ـ المزايا والمواصفات ـ اساليب
الصيانة والعمل جميع الارتباطات الدعائية ـ البرامج المنسوخة والالعاب.
</top>

<top>
<numb> Number: 75
<title>مثلث برمودا
<desc> Description:
ما هو سر مثلث برمودا
<narr> Narrative:
الوثائق التي تحتوي علي معلومات عن مثلث برمودا. قصص اختفاء الطائرات
والسفن في هذ المثلث. المقالات التي تتحدث عن جزيرة برمودا ليس لها علاقة بالموضوع.
</top>

<top>
<numb> Number: 76
<title>قضية استنساخ النعجة دولي
<desc> Description:
كيفية استنساخ النعجة دولي
<narr> Narrative:
خطوات عملية الاستنساخ ومعلومات حول نجاح هذه العملية مع النعجة دولي.
ما ليس له علاقة هو ما يتحدث عن الاستنساخ عموما وراي علماء الدين و استنساخ حيوانات اخرى.
</top>

<top>
<numb> Number: 77
<title>قضية اعتقال موظف الاف بي آي روبرت هنسن
<desc> Description:
كيف تم اكتشاف روبيرت هنسن كجاسوس في الاف بي آي
<narr> Narrative:
كل ما يتعلق باكتشاف روبرت هنسن كجاسوس للمخابرات الروسية.
ما يتعلق بالحرب الباردة بين روسيا وامريكيا في السنوات السابقة ليس لها علاقة بالموضوع
</top>

<top>
<numb> Number: 78
<title>حماس تفجر ديسكو في تل ابيب في فلسطين
<desc> Description:
كيف تم اختراق الحاجز الامني الاسرائيلي وتفجير الديسكو بواسطة فرد من حماس
<narr> Narrative:
الصعوبات التي تواجهها إسرائيل لمنع اختراق الحاجز الامني وتنفيذ
العمليات الانتحارية او الاستشهادية في الاراضي المحتلة. عدد القتلي
والجرحى في هذا الديسكو. العمليات التي تقوم بها بعض الحركات الاخرى
في فلسطين وردود الفعل الاسرائيلية بضرب الاراضي الفلسطينية ليس له علاقة بموضوع البحث.
</top>

<top>
<numb> Number: 79
<title>زلزال السلفادور ٢٠٠١
<desc> Description:
حجم الاضرار والخسائر التي خلفها زلزال السلفادور
<narr> Narrative:
عدد الموتى والذين اصبحوا بدون مأوى نتيجة الزلزال، المباني
والطرق التي دمرها الزلزال. قوة الزلزال والزلازل التي تعرضت لها
السلفادور سابقا كذلك الدول الاخرى في هذه الحقبة نتائج اضافية لما اريد البحث فيه.
</top>

<top>
<numb> Number: 80
<title>مقتل حوالي ٣٠٠٠ شخص في مبنى التجارة العالمي في نيويورك بعد انهياره
<desc> Description:
اسباب انهيار مبنى التجارة العالمي بعد ارتطام الطائرات بالمبني
<narr> Narrative:
الوثائق المرغوب بها هي تلك التي تتحدث عن اسباب انهيار المبنى.
النتائج التي ليس لها علاقة بالموضوع مثل منفذ هذه التفجيرات،
ردود الفعل العالمية عامة والامريكية خاصة بضرب حركة طالبان.
</top>

<top>
<numb> Number: 81
<title>قضية الاطفال الليبين المصابون بفيروس الايدز
<desc> Description:
متى وكيف تم حقن هؤلاء الاطفال بالفيروس، ومن المستفيد من هذا الفعل
<narr> Narrative:
الوثائق التي تتحدث عن كيف تم حقن الاطفال بفيروس الايدز. ومن المستفيد من هذه العملية.
</top>

<top>
<numb> Number: 82
<title>الحكم في قضية لوكوربي
<desc> Description:
كيف كانت ردود الفعل العربية من ادانة احد المتهمين الليبين في قضية البانام ١٠٣.
<narr> Narrative:
ردود الفعل الايجابية والسلبية من قبل الدول العربية حول ادانة
احد الليبيين في قضية لوكربي. المقالات التي تتحدث عن تفاصيل
الحادثة في ١٩٨٦ غير مرغوب فيها
</top>

<top>
<numb> Number: 83
<title>كسوف الشمس
<desc> Description:
كيف يحدث كسوف الشمس؟
<narr> Narrative:
اسباب كسوف الشمس، انواع كسوف الشمس ، تواريخ حدوث كسوف الشمس
</top>

<top>
<numb> Number: 84
<title>الانتخابات البرلمانية في كوستريكا
<desc> Description:
من هو الحزب الفائز
<narr> Narrative:
المقالات المتوقعة هي كل ما يتحدث عن الاحزاب في كوستريكا، ومن
هو الحزب الحاكم. المقالات التي تتحدث عن ردود الفعل الدولية
والمحلية حول نتائج الانتخابات غير مرغوب فيها
</top>

<top>
<numb> Number: 85
<title>الانتخابات الالمانية
<desc> Description:
نتائج الانتخابات الالمانية
<narr> Narrative:
المقالات المتوقعة هي تلك التي تتحدث عن فوز المستشار السابق جيرهادر شرويدر في الانتخابات الالمانية.
</top>

<top>
<numb> Number: 86
<title>عملة اوروبا
<desc> Description:
ما هي العملة الاوربية الواحدة
<narr> Narrative:
كل ما يتحدث عن اعلان اليورو كعملة اوربا. ما يتحدث حول السوق
المالية والاسهم والسندات ليس له علاقة بالموضوع كذلك ما يتعلق
بالعملات الاخرى ين، دولار ، استرليني، فرنك
</top>

<top>
<numb> Number: 87
<title>سلاح العراق النووي
<desc> Description:
من هو رئيس لجنة الكشف على سلاح العراق النووي
<narr> Narrative:
المقالات المتوقعة هى التي تحتوي على اسم رئيس اللجنة محمد
البرادعي ، هانز بليكس ومعلومات عنهما.
</top>

<top>
<numb> Number: 88
<title>فرنسا وموناكو
<desc> Description:
ما هو موقف فرنسا من قضية موناكو؟
<narr> Narrative:
المقالات التي تتحدث عن الصراع بين فرنسا وموناكو كدولة.
رد فعل فرنسا من مطالبة موناكو بالاستقلال. موقف فرنسا من قضية موناكو.
</top>

<top>
<numb> Number: 89
<title>فضيحة ابوغريب
<desc> Description:
من وراء فضيحة سجن ابوغريب وكيف تم كشفها؟
<narr> Narrative:
كافة المقالات التي تصف الممارسات الخاظئة التي قام بها الجنود
الامريكان في سجن ابوغريب . ومن المسؤول عنها ومن كشف هذه الممارسات.
المقالات التي تتحدث عن ردود الفعل وتصريحات المسؤولين
الامريكيين حول الحادثة ليس لها علاقة.
</top>

<top>
<numb> Number: 90
<title>اصابة الرئيس الامريكي السابق رولاند ريقن بمرض الزهايمر
<desc> Description:
ماهو رد فعل السياسين الامريكين بعد اصابة رولاند ريقن بمرض الزهايمر
<narr> Narrative:
كل ردود الفعل للسياسين الامريكين هو النتيجة المطلوبة
اماردود الفعل لغير الامريكين غير مهم
</top>

T.No.	All	NR	R	T.No.	All	NR	R	T.No.	All	NR	R
1	221	215	6	31	218	151	67	61	177	169	8
2	225	201	24	32	230	206	24	62	169	158	11
3	241	191	50	33	512	479	33	63	205	153	52
4	276	168	108	34	400	364	36	64	168	113	55
5	296	200	96	35	403	367	36	65	274	247	27
6	183	150	33	36	289	276	13	66	258	248	10
7	208	185	23	37	378	318	60	67	293	267	26
8	352	331	21	38	416	373	43	68	200	174	26
9	355	261	94	39	217	117	100	69	203	195	8
10	328	320	8	40	383	302	81	70	192	180	12
11	107	82	25	41	149	128	21	71	557	537	20
12	148	136	12	42	219	213	6	72	198	136	62
13	634	554	80	43	209	38	171	73	187	183	4
14	208	112	96	44	146	139	7	74	429	386	43
15	356	334	22	45	403	354	49	75	129	127	2
16	451	401	50	46	131	104	27	76	153	115	38
17	164	135	29	47	283	264	19	77	445	424	21
18	585	563	22	48	187	86	101	78	154	147	7
19	363	344	19	49	117	101	16	79	119	68	51
20	301	262	39	50	594	416	178	80	170	163	7
21	262	222	40	51	326	317	9	81	137	120	17
22	455	301	154	52	334	318	16	82	176	172	4
23	162	28	134	53	464	388	76	83	235	185	50
24	265	182	83	54	141	120	21	84	605	598	7
25	369	355	14	55	272	227	45	85	392	338	54
26	475	306	169	56	473	466	7	86	383	258	125
27	116	61	55	57	173	162	11	87	262	226	36
28	233	173	60	58	308	302	6	88	366	359	7
29	266	156	110	59	193	142	51	89	509	390	119
30	278	265	13	60	226	146	80	90	359	331	2

Table A.1: Topic numbers and their respective number of annotated documents. "T.No." stands for topic no, "All" stands for the total number of annotated documents per each topic, "NR" stands of the number of non-relevant documents, and "R" stands for the number of relevant documents. The average documents annotated per topic is 286.5, the average number of non-relevant documents per topic is 241.6, and the average number of relevant documents per topic is 44.8.

Appendix B

Foreign Words Expansion Results

This appendix shows results obtained by experiments in Chapter 7. The performance of the light11 stemmer using similarity algorithms to expand foreign words in the query is shown here. Tables show the performance of the stemmer in terms of Recall, P@10, and R-Precision for expanding both automatically and manually identified foreign words in the queries.

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM	0.2539	0.2607	0.2652	0.2640	0.2652	0.2652	0.2652	0.2652
NORM1	0.2494	0.2551	0.2533	0.2689	0.2656	0.2611	0.2611	0.2622
NORM2	0.2494	0.2584	0.2562	0.2562	0.2562	0.2562	0.2562	0.2562
NORM3	0.2494	0.2584	0.2562	0.2562	0.2562	0.2562	0.2562	0.2562
gramCount	0.2478	0.2467	0.2422	0.2500	0.2567	0.2556	0.2567	0.2578
gramDist	0.2444	0.2444	0.2389	0.2511	0.2500	0.2500	0.2478	0.2311
LCS	0.2500	0.2589	0.2667	0.2667	0.2678	**0.2756**	0.2744	**0.2689**
Sgrams	0.2389	0.2389	0.2456	0.2500	0.2544	0.2567	0.2567	0.2456
Asoundex-Final	0.2528	0.2539	0.2506	0.2506	0.2506	0.2506	0.2506	0.2506
Soutex	0.2528	0.2539	0.2611	0.2678	0.2667	0.2633	0.2644	0.2622
Soutex4	0.2494	0.2551	0.2544	**0.2700**	0.2678	0.2711	0.2733	0.2644
AEditex	0.2489	0.2556	**0.2700**	0.2622	**0.2722**	0.2689	**0.2678**	0.2656
REditex	**0.2629**	**0.2678**	0.2667	0.2689	0.2689	0.2722	0.2722	0.2667
Dice	0.2478	0.2467	0.2422	0.2500	0.2567	0.2533	0.2544	0.2433
EditDistance	0.2444	0.2589	0.2667	0.2656	0.2689	0.2678	0.2667	0.2678

Table B.1: The **P@10** *scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. The baseline is the light11 stemmer (P@10=0.2533). Foreign words expanded are those* **automatically** *identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer.*

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM	0.5907	0.5917	0.5998	0.5998	0.6061	0.6064	0.6064	0.6064
NORM1	0.5917	0.5879	0.5936	0.6114	0.6110	0.6105	0.6102	0.6075
NORM2	0.5907	0.5960	0.5978	0.5980	0.5980	0.5980	0.5980	0.5980
NORM3	0.5907	0.5960	0.5978	0.5980	0.5980	0.5980	0.5980	0.5980
gramCount	0.5691↓	0.5681↓	0.5721↓	0.5879↓	0.5986	0.5981	0.5968	0.6087
gramDist	0.5691↓	0.5681↓	0.5716↓	0.5911↓	0.5993↓	0.5991↓	0.5968↓	0.5884↓
LCS	0.5708	0.5668	0.5830	0.5859	0.5847	0.6174	0.6137	0.6350
Sgrams	0.5716↓	0.5711↓	0.5716↓	0.5996↓	0.6005↓	0.5929↓	0.5926↓	0.5896↓
Asoundex-Final	0.5907	0.5856	0.5894	0.5922	0.5901	0.5901	0.5901	0.5901
Soutex	**0.6011**	**0.5983**	0.5849	0.6045	0.6050	0.6050	0.6100	0.6062
Soutex4	0.5993	0.5965	**0.6053**	**0.6216**	**0.6248**	**0.6318**	**0.6598**	**0.6563**
AEditex	0.5683↓	0.5711↓	0.5792	0.5750	0.6062	0.6092	0.6080	0.6090
REditex	0.5909	0.5792↓	0.5790↓	0.5797↓	0.5792↓	0.5802↓	0.5792↓	0.5790↓
Dice	0.5691	0.5681↓	0.5721↓	0.5879	0.5986	0.5976	0.5961	0.6070
EditDistance	0.5698↓	0.5713	0.5753	0.5748	0.5998	0.6067	0.6062	0.6169

Table B.2: The **Recall** *scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. the baseline is running the light11 stemmer without query expansion (Recall=0.6102). Foreign words expanded are those* **automatically** *identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer.*

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM	**0.2034**	**0.2104**	**0.2137**	0.2130	0.2147	0.2147	0.2147	0.2147
NORM1	0.2004	0.2090	0.2108	0.2185	0.2185	0.2176	0.2166	0.2179
NORM2	0.1997	0.2074	0.2081	0.2076	0.2076	0.2076	0.2076	0.2076
NORM3	0.1997	0.2074	0.2076	0.2076	0.2076	0.2076	0.2076	0.2076
gramCount	0.1932	0.1908	0.1918	0.1964	0.2004	0.1970	0.1943	0.1935
gramDist	0.1865	0.1850	0.1837↓	0.1935	0.1885	0.1861	0.1827	0.1773
LCS	0.1948	0.1988	0.2079	0.2095	0.2087	0.2157	0.2151	0.2118
Sgrams	0.1854↓	0.1865	0.1894	0.1941	0.1938	0.1930	0.1915	0.1905
Asoundex-Final	0.2031	0.2043	0.2042	0.2044	0.2044	0.2044	0.2044	0.2044
Soutex	0.1992	0.2017	0.2098	0.2159	0.2137	0.2128	0.2147	0.2129
Soutex4	0.2011	0.2070	0.2105	**0.2214**	**0.2200**	**0.2205**	**0.2251↑**	**0.2208**
AEditex	0.1882	0.1997	0.2097	0.2044	0.2101	0.2102	0.2090	0.2029
REditex	0.1994	0.2089	0.2075	0.2082	0.2079	0.2116	0.2127	0.2085
Dice	0.1928	0.1908	0.1918	0.1964	0.2004	0.1963	0.1935	0.1845
EditDistance	0.1869	0.1992	0.2054	0.2050	0.2102	0.2134	0.2117	0.2065

Table B.3: The **R-Precision** *scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. The baseline is running the light11 stemmer without query expansion (R-Precision=0.2003). Foreign words expanded are those* **automatically** *identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer, while ↑ indicates results that are significantly better than the light11 stemmer in the 95% confidence level.*

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM	0.2584	0.2562	0.2539	0.2562	0.2551	0.2551	0.2528	0.2539
NORM1	0.2467	0.2567	**0.2700**	0.2722	0.2689	0.2678	0.2667	0.2656
NORM2	0.2517	0.2528	0.2506	0.2506	0.2506	0.2506	0.2506	0.2506
NORM3	0.2483	0.2494	0.2483	0.2483	0.2483	0.2483	0.2483	0.2483
gramCount	0.2378	0.2367	0.2400	0.2467	0.2600	0.2522	0.2522	0.2456
gramDist	0.2483	0.2444	0.2478	0.2611	0.2533	0.2500	0.2433	0.2278
LCS	0.2444	0.2533	0.2622	0.2622	0.2644	0.2744	0.2756	0.2656
Sgrams	0.2322	0.2333	0.2389	0.2500	0.2489	0.2511	0.2511	0.2400
Asoundex-Final	0.2467	0.2467	0.2422	0.2411	0.2400	0.2400	0.2400	0.2400
Soutex	0.2611	0.2644	0.2667	0.2756↑	0.2756↑	0.2756↑	**0.2778**↑	**0.2722**↑
Soutex4	0.2400	0.2444	0.2611	**0.2778**	**0.2789**	**0.2822**	0.2733	0.2611
AEditex	0.2378	0.2489	0.2656	0.2567	0.2656	0.2622	0.2611	0.2589
REditex	0.2596	**0.2678**	0.2667	0.2667	0.2656	0.2689	0.2689	0.2644
Dice	0.2400	0.2389	0.2333	0.2456	0.2578	0.2556	0.2522	0.2367
EditDistance	**0.2633**	0.2633	0.2633	0.2633	0.2633	0.2633	0.2633	0.2633

Table B.4: The **P@10** *scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. The baseline is the light11 stemmer without query expansion (P@10=0.2533). Foreign words expanded are those* **manually** *identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer, while ↑ indicates results that are significantly better than the light11 stemmer in the 95% confidence level.*

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM	0.5607	0.5652	0.5741	0.5851	0.5846	0.5838	0.5830	0.5828
NORM1	0.5609↓	0.5760	**0.6015**	0.6005	0.6000	0.5976	0.5968	0.5931
NORM2	0.5642	0.5746	0.5734	0.5752	0.5752	0.5752	0.5752	0.5752
NORM3	0.5635	0.5739	0.5729	0.5744	0.5744	0.5744	0.5744	0.5744
gramCount	0.5530↓	0.5525↓	0.5574↓	0.5906	0.6127	0.6191	0.6300	0.6258
gramDist	0.5774	0.5626↓	0.5777↓	**0.6256**	0.6184	0.6184	0.6080	0.6005↓
LCS	0.5555	0.5515↓	0.5681	0.5723	0.5713	0.6038	0.5991	0.6164
Sgrams	0.5584↓	0.5594↓	0.5656↓	0.6117	0.6134	0.6124	0.6122	0.6070↓
Asoundex-Final	0.5545↓	0.5537↓	0.5515↓	0.5498↓	0.5498↓	0.5495↓	0.5493↓	0.5493↓
Soutex	0.5805	0.5810	0.5859	0.5958	0.5961	0.5963	0.6013	0.6008
Soutex4	0.5520↓	0.5716	0.5857	0.6010	**0.6290**	**0.6315**	**0.6305**	0.6258
AEditex	0.5381↓	0.5530↓	0.5646	0.5597	0.5901	0.5958	0.5948	0.6018
REditex	0.5861	0.5785↓	0.5795↓	0.5800↓	0.5787	0.5800	0.5795	0.5743↓
Dice	0.5527↓	0.5517↓	0.5584↓	0.5820	0.6119	0.6110	0.6181	0.6263
EditDistance	**0.5867**↓	**0.5867**↓	0.5867↓	0.5867↓	0.5867↓	0.5867↓	0.5867↓	0.5867↓

Table B.5: The **Recall** *scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. The baseline is running the light11 stemmer without query expansion (Recall=0.6102). Foreign words expanded are those* **manually** *identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer.*

	Number of variants used in query expansion							
Expanded With	3	5	10	20	30	40	50	100
NORM	0.1949	0.1966	0.1966	0.1977	0.1978	0.1967	0.1948	0.1945
NORM1	0.1980	**0.2083**	**0.2146**	**0.2174**	0.2160	0.2142	0.2129	0.2117
NORM2	0.1914	0.1934	0.1914	0.1909	0.1909	0.1909	0.1909	0.1909
NORM3	0.1899	0.1921	0.1907	0.1896	0.1896	0.1896	0.1896	0.1896
gramCount	0.1882	0.1866	0.1896	0.1926	0.1963	0.1905	0.1898	0.1884
gramDist	0.1881	0.1844↓	0.1901	0.1962	0.1872	0.1871	0.1845	0.1773
LCS	0.1956	0.1996	0.2099	0.2087	0.2062	0.2134	**0.2149**	**0.2118**
Sgrams	0.1816↓	0.1831↓	0.1860	0.1917	0.1889	0.1887	0.1873↓	0.1866↓
Asoundex-Final	0.1941	0.1907	0.1878	0.1879	0.1874	0.1872	0.1871	0.1871
Soutex	0.1988	0.2029	0.2060	0.2112	0.2096	0.2094	0.2094	0.2061
Soutex4	0.1897	0.1961	0.2091	0.2145	**0.2167**	**0.2184**	0.2143	0.2065
AEditex	0.1828↓	0.1980	0.2107	0.2042	0.2098	0.2079	0.2099	0.1990
REditex	0.1945	0.2078	0.2057	0.2064	0.2044	0.2071	0.2096	0.2042
Dice	0.1918	0.1893	0.1894	0.1933	0.1962	0.1922	0.1879	0.1781
EditDistance	**0.2009**	0.2009	0.2009	0.2009	0.2009	0.2009	0.2009	0.2009

Table B.6: The **R-Precision** *scores of the light11 stemmer when expanding queries using the top 3, 5, 10, 20, 30, 50, and 100 variants returned by similarity matching algorithms. The baseline is running the light11 stemmer without query expansion (RP=0.2003). Foreign words expanded are those* **manually** *identified as foreign in queries. ↓ indicates results that are significantly worse than the light11 stemmer.*

Bibliography

A. Abdelali, J. Cowie, and H. S. Soliman. Arabic information retrieval perspectives. In *Proceedings of the 11th Conference on Natural Language Processing, Journes d'Etude sur la Parole - Traitement Automatique des Langues Naturelles (JEP-TALN)*, Fez, Morocco, April 2004.

N. Abduljaleel and L. Larkey. English to Arabic transliteration for information retrieval: A statistical approach. Technical Report IR-261, Univeristy of Massachusetts, 2002.

N. Abduljaleel and L. S. Larkey. Statistical transliteration for English-Arabic cross-language information retrieval. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 139–146, New Orleans, LA, 2003. ACM Press. ISBN 1-58113-723-0.

H. Abu-Salem. *A Microcomputer Based Arabic Information Retrieval System with Relational Thesauri (Arabic-IRS).* PhD thesis, Illinois Institute of Technology, Chicago, IL, August 1992.

H. Abu-Salem, M. Al-Omari, and M. W. Evens. Stemming methodologies over individual query words for an Arabic information retrieval system. *Journal of the American Society for Information Science*, 50(6):524–529, 1999. ISSN 0002-8231.

H. K. Al Ameed, S. O. Al Ketbi, A. A. Al Kaabi, K. S. Al Shebli, N. F. Al Shamsi, N. H. Al Nuaimi, and S. S. Al Muhairi. Arabic light stemmer: A new enhanced approach. In *Proceedings of The Second International Conference on Innovations in Information Technology (IIT'05)*, pages 1–9, Dubai, UAE, 26–28 September 2005.
URL: http://www.it-innovations.ae/iit005/proceedings/articles/G_1_IIT05_Hayder.pdf.

S. S. Al-Fedaghi and F. Al-Anzi. A new algorithm to generate Arabic root-pattern forms.

In *Proceedings of the 11th National Computer Conference and Exhibition*, pages 391–400, Dhahran, Saudi Arabia, March 1989.

I. A. Al-Kharashi. *Micro-AIRS: a microcomputer-based Arabic information retrieval system comparing words, stems, and roots as index terms.* PhD thesis, Illinois Institute of Technology, Chicago, IL, 1991.

I. A. Al-Kharashi and M. W. Evens. Comparing words, stems, and roots as index terms in an Arabic information retrieval system. *Journal of the American Society for Information Science*, 45(8):548–560, 1994.

A. Al-Maskari, M. Sanderson, and P. Clough. The relationship between IR effectiveness measures and user satisfaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 773–774, Amsterdam, The Netherlands, 2007. ACM Press. ISBN 978-1-59593-597-7.

J. B. S. Al-Qinal. Morphophonemics of loanwords in translation. *Journal of King Saud University*, 13:1–132, 2002.

R. Al-Shalabi and M. Evens. A computational morphology system for Arabic. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages (COLING-ACL '98)*, pages 66–72, Montreal, Quebec, Canada, 16 August 1998.
URL: http://www.cs.um.edu.mt/~mros/casl/prog.html.

M. S. Al-Shanti. *Al Maharat Allughawia.* Al Andalus for publishing and distribution, fourth edition, 1996. In Arabic.

I. A. Al-Sughaiyer and I. A. Al-Kharashi. Arabic morphological analysis techniques: A comprehensive survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213, 2004.

M. Alghamdi. Algorithms for romanizing Arabic names. *Journal of King Saud University: Computer Sciences and Information.*, 17:1–27, 2005. In Arabic.

M. Aljlayl and O. Frieder. Effective Arabic-English cross-language information retrieval via machine-readable dictionaries and machine translation. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 295–302, Atlanta, Georgia, 2001. ACM Press. ISBN 1-58113-436-3.

M. A. Aljlayl. *On Arabic Search: The Effectiveness of Monolingual and Bidirectional Information Retrieval*. PhD thesis, The Graduate College of the Illinois Institute of Technology, 2002.

M. A. Aljlayl and O. Frieder. On Arabic search: improving the retrieval effectiveness via a light stemming approach. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 340–347, McLean, Virginia, 2002. ACM Press. ISBN 1-58113-492-4.

A. M. AlShehri. *Optimization and effectiveness of n-grams approach for indexing and retrieval in Arabic information retrieval systems*. PhD thesis, School of Information Science, University of Pittsburgh, 2002.

S. U. Aqeel, S. Beitzel, E. Jensen, D. Grossman, and O. Frieder. On the development of name search techniques for Arabic. *Journal of the American Society for Information Science and Technology*, 57(6):728–739, 2006. ISSN 1532-2882.

M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart. Algorithms for Arabic name transliteration. *IBM Journal of Research and Development*, 38(2):183–194, 1994. ISSN 0018-8646.

J. Asian. *Effective Techniques for Indoneian Text Retrieval*. PhD thesis, School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, 2007.

M. Attia. Developing a robust Arabic morphological transducer using finite state technology. In *Proceedings of the 8th Annual UK special-interest group for computational linguistics (CLUK) research Colloquium*, University of Manchester, UK, 2005.

M. Attia. An ambiguity-controlled morphological analyzer for modern standard Arabic modelling finite state networks. In *Proceedings of the Challenge of Arabic for NLP/MT Conference*. The British Computer Society, London, October 2006.

M. Attia. Arabic tokenization system. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 65–72, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
URL: http://www.aclweb.org/anthology/W/W07/W07-0809.

R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN 0-201-39829-X.

BBC News. Social networks top Google search, 18 December 2006.
URL: http://news.bbc.co.uk/go/pr/fr/-/2/hi/technology/6189809.stm, [Online; accessed 26 Febreuary 2008].

K. R. Beesley. Arabic finite-state morphological analysis and generation. In *Proceedings of the 16th International Conference on Computaional Linguistics (COLING '96)*, volume 1, pages 89–94, Copenhagen, Denmark, 1996. Association for Computational Linguistics.

K. R. Beesley. Computer analysis of Arabic morphology: A two-level approach with detours. In B. Comrie and M. Eid, editors, *Proceedings of the Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics*, volume 12, pages 155–172. John Benjamins, Amsterdam, 1991.

K. R. Beesley. Arabic morphological analysis on the Internet. In *Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing*, Cambridge University, UK, 17-18 April 1998.

K. R. Beesley, T. Buckwalter, and S. N. Newton. Two-level finite-state analysis of Arabic morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*, Cambridge, England, 6–7 September 1989.

Y. Benajiba, P. Rosso, and J.-M. Benedí. Anersys: An Arabic named entity recognition system based on maximum entropy. In A. F. Gelbukh, editor, *Proceedings of the 8th International Conference of the Computational Linguistics and Intelligent Text Processing, (CICLing '07)*, volume 4394 of *Lecture Notes in Computer Science*, pages 143–153, Mexico City, Mexico, 18–24 February 2007. Springer. ISBN 3-540-70938-X.

B. Bishop. A history of the Arabic language, 24 April 1998.
URL: http://linguistics.byu.edu/classes/ling450ch/reports/arabic.html.

C. L. Borgman and S. L. Siegfried. Getty's synoname and its cousins: A survey of applications of personal name-matching algorithms. *Journal of the American Society for Information Science*, 43(7):459–476, 1992.

C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, pages 292–300, Dublin, Ireland, 1994. Springer-Verlag New York, NY. ISBN 0-387-19889-X.

T. Buckwalter. Buckwalter Arabic morphological analyzer version 1.0, 2002. LDC Catalog No. LDC2002L49.

T. Buckwalter. Buckwalter Arabic morphological analyzer version 2.0, 2004. LDC Catalog No. LDC2004L02.

J. Callan, W. B. Croft, and S. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Application*, pages 78–83, 1992.

J. P. Callan, W. B. Croft, and J. Broglio. TREC and TIPSTER experiments with INQUERY. *Information Processing & Management*, 31(3):327–343, 1995. ISSN 0306-4573.

W. B. Cavnar and J. M. Trenkle. N-gram based text categorization. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR '94)*, pages 161–175, Las Vegas, US, April 1994.

J. Celko. *Joe Celko's SQL for smarties: advanced SQL programming.* Morgan Kaufmann Publishers Inc., Amsterdam, Boston, third edition, 2005.

S.-F. Chang, W. Hsu, L. Kennedy, L. Xie, A. Yanagawa, E. Zavesky, and D. Zhang. Columbia university TRECVID-2005 video search and high-level feature extraction. In *Proceedings of the TRECVID 2005 Workshop*, Gaithersburg, Maryland, November 2005.

A. Chen and F. Gey. Building an Arabic stemmer for information retrieval. In *NIST Special Publication 500-251: Proceedings of the Eleventh Text REtrieval Conference (TREC 2002).* National Institute of Standards and Technology, November 2002.
URL: http://trec.nist.gov/pubs/trec11/papers/ucalberkeley.chen.pdf.

A. Chowdhury, M. C. McCabe, D. Grossman, and O. Frieder. Document normalization revisited. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 381–382. ACM Press, 2002. ISBN 1-58113-561-0.

G. G. Chowdhury. *Introduction to modern information retrieval.* Facet Publishing, London, UK, second edition, 2004.

P. Christen. A comparison of personal name matching: Techniques and practical issues. Technical Report TR-CS-06-02, The Australian National University, September 2006a.

P. Christen. A comparison of personal name matching: Techniques and practical issues. In *Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pages 290–294, Los Alamitos, CA, 2006b. IEEE Computer Society. ISBN 0-7695-2702-7.

G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 282–289, Melbourne, Australia, 1998. ACM Press. ISBN 1-58113-015-5.

A. Corrada-Emmanuel and W. B. Croft. Answer models for question answering passage retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*, pages 516–517, Sheffield, UK, 2004. ACM Press. ISBN 1-58113-881-4.

F. J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964. ISSN 0001-0782.

K. Darwish and D. W. Oard. Term selection for searching printed Arabic. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 261–268, Tampere, Finland, 2002. ACM Press. ISBN 1-58113-561-0.

K. Darwish and D. W. Oard. Probabilistic structured query methods. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*, pages 338–344, Toronto, Canada, 2003a. ACM Press. ISBN 1-58113-646-3.

K. Darwish and D. W. Oard. CLIR experiments at Maryland for TREC 2002: Evidence combination for Arabic-English retrieval. Technical Report LAMP-TR-101,CS-TR-4456,UMIACS-TR-2003-26, University of Maryland, College Park, February 2003b.
URL: http://lampsrv01.umiacs.umd.edu/pubs/TechReports/LAMP_101/LAMP_101.pdf.

K. Darwish and D. W. Oard. *Adapting Morphology for Arabic Information Retrieval*, volume 38 of *Text, Speech and Language Technology*, pages 245–262. Springer Netherlands, 2007. ISBN 978-1-4020-6045-8.

K. Darwish, D. Doermann, R. Jones, D. Oard, and M. Rautiainen. TREC 10 experiments at university of Maryland clir and video. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 549–561, Gaithersburg, Maryland, 2001.

K. Darwish, H. Hassan, and O. Emam. Examining the effect of improved context sensitive morphology on Arabic information retrieval. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 25–30, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.
URL: http://www.aclweb.org/anthology/W/W05/W05-0704.

S. Dennis, R. McArthur, and P. Bruza. Searching the World Wide Web made easy? the cognitive load imposed by query re nement mechanisms. In *Proceedings of the 3rd Australian Document Computing Symposium*, pages 65–71, Sydney, Australia, 1998.

T. DeYoung. Arabic language: Background and history, 1999.
URL: http://www.iub.edu/~arabic/Summer2004/Arabic_History.pdf.

M. Diab, K. Hacioglu, and D. Jurafsky. Automatic tagging of Arabic text: from raw text to base phrase chunks. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL): Short Papers*, pages 149–152, Boston, Massachusetts, 2–7 May 2004. Association for Computational Linguistics.

L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3): 297–302, July 1945.

S. Doraisamy and S. Rüger. Robust polyphonic music retrieval with n-grams. *Journal of Intelligent Information Systems*, 21(1):53–70, 2003. ISSN 0925-9902.

T. Dunning. Statistical identification of language. Technical Report MCCS-94-273, Computing Research Lab (CRL), New Mexico State University, 1994.

I. H. A. El-Khair. *Effectiveness of document processing techniques for Arabic information retrieval*. PhD thesis, School of Information Science, University of Pittsburgh, 2003.

T. A. El-Sadany and M. A. Hashish. An Arabic morphological system. *IBM Systems Journal*, 28(4):600–612, 1989. ISSN 0018-8670.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 1–8, Boston, MA, 2–7 May 2004. Association for Computational Linguistics.

C. Foley, C. Gurrin, G. Jones, H. Lee, S. McGivney, N. E. O'Connor, S. Sav, A. F. Smeaton, and P. Wilkins. TRECVid 2005 experiments at dublin city university. In *Proceedings of the TRECVID 2005 Workshop*, Gaithersburg, Maryland, 14–15 November 2005.

W. B. Frakes and R. A. Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992. ISBN 0-13-463837-9.

W. B. Frakes and C. J. Fox. Strength and similarity of affix removal stemming algorithms. *SIGIR Forum*, 37(1):26–30, 2003. ISSN 0163-5840.

T. Gadd. Phonix: the algorithm. *Program*, 24(4):363–369, 1990.

F. C. Gey and D. W. Oard. The TREC 2001 cross-language information retrieval track: Searching Arabic using English , French or Arabic queries. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 16–25, Gaithersburg, Maryland, 2001.

Y. Goldberg and M. Elhadad. Identification of transliterated foreign words in Hebrew script. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '08)*, Lecture Notes in Computer Science, pages 466–477, Haifa, Israel, 17–23 February 2008. Springer, Heidelberg, Germany. ISBN 978-3-540-78134-9.

R. Gong and T. K. Chan. Syllable alignment: A novel model for phonetic string search. *IEICE transactions on information and systems*, 89(1):332–339, 2006. ISSN 0916-8532.

A. Goweder and A. D. Roeck. Assessment of a significant Arabic corpus. In *Proceedings of the ACL/EACL 2001 Workshop on Arabic Natural Language Processing: Status and Prospects*, Toulouse, France, 6 July 2001.
URL: http://www.elsnet.org/arabic2001/goweder.pdf.

A. Goweder, M. Poesio, and A. D. Roeck. Broken plural detection for Arabic information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval (SIGIR '04), pages 566–567, Sheffield, UK, 2004. ACM Press. ISBN 1-58113-881-4.

D. Graff, K. Chen, J. Kong, and K. Maeda. Arabic Gigaword second edition. Linguistic Data Consortium, January 2006. LDC Catalog No. LDC2006T02.

D. Grune and C. Jacobs. *Parsing Techniques A Practical Guide*. Ellis Horwood limited, Amsterdam, Netherlands, first edition edition, 1994. ISBN 0-13-651431-6.
URL: ftp://ftp.cs.vu.nl/pub/dick/PTAPG_1st_Edition/BookBody.pdf.

N. Habash. Large scale lexeme based Arabic morphological generation. In *Proceedings of the 11th Conference on Natural Language Processing, Journes d'Etude sur la Parole - Traitement Automatique des Langues Naturelles (JEP-TALN)*, pages 271–276, Fez, Morocco, April 2004.

N. Habash. *Arabic Morphological Representations for Machine Translation*, volume 38 of *Text, Speech and Language Technology*, pages 263–285. Springer Netherlands, 2007. ISBN 978-1-4020-6045-8.

N. Habash and O. Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*, pages 573–580, University of Michigan, Michigan, 25–30 June 2005. Association for Computational Linguistics.

P. A. V. Hall and G. R. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, 1980. ISSN 0360-0300.

J. Halpern. The challenges and pitfalls of Arabic Romanization and Arabization. In *Proceedings of the Second Workshop on Computational Approaches to Arabic Script-based Languages*, pages 47–54, Stanford, CA, 21–22 July 2007.

S. M. Harding, W. B. Croft, and C. Weir. Probabilistic retrieval of OCR degraded text using n-grams. In *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries (ECDL '97)*, pages 345–359, London, UK, 1997. Springer-Verlag. ISBN 3-540-63554-8.

D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7–15, 1991.

B. He and I. Ounis. Term frequency normalisation tuning for bm25 and dfr models. In *Proceedings of the 27th European Conference on IR Research (ECIR '05)*, volume 3408 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2005. ISBN 3-540-25295-9.

M. R. Henzinger. Tutorial 1: Web information retrieval. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, page 693, 2000.
URL: http://www.tcnj.edu/~mmmartin/CMSC485/Papers/Google/icde.pdf, Presentation slides.

I. Hmeidi, G. Kanaan, and M. Evens. Design and implementation of automatic indexing for information retrieval with Arabic documents. *Journal of the American Society for Information Science*, 48(10):867–881, 1997. ISSN 0002-8231.

D. Holmes and M. C. McCabe. Improving precision and recall for soundex retrieval. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '02)*, pages 22–27, Los Alamitos, CA, 2002. IEEE Computer Society. ISBN 0-7695-1506-1.

D. Holmes, S. Kashfi, and S. U. Aqeel. Transliterated Arabic name search. In *Proceedings of the 3rd International Conference on Communications, Internet, and Information Technology (IASTED '03)*, pages 267–273, St. Thomas, US Virgin Islands, 2004.

W. H. Hsu and S.-F. Chang. Visual cue cluster construction via information bottleneck principle and kernel density estimation. In *Proceedings of the 4th International Conference on Image and Video Retrieval (CIVR)*, pages 82–91, Singapore, 20-22 July 2005.

W. H. Hsu, L. Kennedy, S.-F. Chang, M. Franz, and J. Smith. Columbia-IBM news video story segmentation in TRECVID 2004. Technical Report 209-2005-3, Columbia ADVENT, New York, 2005.

D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*, pages 329–338, Pittsburgh, Pennsylvania, 1993. ACM Press.

D. Hull. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996.

D. Hull and G. Grefenstette. Querying across languages: a dictionary-based approach to multilingual information retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 49–57, Zurich, Switzerland, 1996. ACM Press.

IPA. *Handbook of the International Phonetic Association*. Cambridge University Press, Cambridge, UK, 1999. ISBN 0-521-65236-7.

B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998. ISSN 0163-5840.

N. Jardine and C. J. van Rijsbergen. The use of hierarchical clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.

K. S. Jeong, S. H. Myaeng, J. S. Lee, and K.-S. Choi. Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing and Management*, 35(4):523–540, 1999.

M. Jiyad. A hundred and one rules !, 2005.
URL: http://www.mtholyoke.edu/courses/mjiyad/ArabicGrammarBook.doc.

K. S. Jones and C. van Rijsbergen. *Report on the need for and provision of an "ideal" information retrieval test collection (British Library Research and Development Report No. 5266)*. Computer Laboratory, University of Cambridge, England, 1975.

Y. Kadri and J.-Y. Nie. Effective stemming for Arabic information retrieval. the challenge of Arabic for NLP/MT. In *Proceedings of the International conference at the British Computer Society*, pages 68–74, London, UK, 23 October 2006.
URL: http://www.mt-archive.info/BCS-2006-Kadri.pdf.

D. Kamir, N. Soreq, and Y. Neeman. A comprehensive NLP system for modern standard Arabic and modern Hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–9, Philadelphia, Pennsylvania, 2002. Association for Computational Linguistics.

B.-J. Kang and K.-S. Choi. Effective foreign word extraction for Korean information retrieval. *Information Processing and Management*, 38(1):91–109, 2002.

M. M. Kashani, F. Popowich, and A. Sarkar. Automatic transliteration of proper nouns from Arabic to English. In *Proceedings of the Second Workshop on Computational Approaches to Arabic Script-based Languages*, Stanford, CA, 21–22 July 2007.

S. Khoja and R. Garside. Stemming Arabic text. Technical report, Computing Department, Lancaster University, Lancaster, September 1999.

G. A. Kiraz. Arabic computational morphology in the West. In *Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing*, Cambridge University, 17-18 April 1998.

P. Koehn and C. Monz. NAACL 2006 workshop on statistical machine translation, June 2006.
URL: http://www.statmt.org/wmt06/, [Online; accessed 26 February 2008].

R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*, pages 191–202, Pittsburgh, Pennsylvania, 1993. ACM Press. ISBN 0-89791-605-0.

K. Kuriyama, N. Kando, T. Nozue, and K. Eguchi. Pooling for a large-scale test collection: An analysis of the search results from the first NTCIR workshop. *Journal of Information Retrieval*, 5(1):41–59, 2002. ISSN 1386-4564.

L. Larkey, N. Abduljaleel, and M. Connell. What's in a name?: Proper names in Arabic cross language information retrieval. Technical Report IR-278, Univeristy of Massachusetts, 2003.

L. S. Larkey and M. E. Connell. Arabic information retrieval at UMass in TREC 10. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 562–570, 2001.
URL: http://trec.nist.gov/pubs/trec10/papers/UMass_TREC10_Final.pdf.

L. S. Larkey and M. E. Connell. Structured queries, language modeling, and relevance modeling in cross-language information retrieval. *Information Processing and Management Special Issue on Cross Language Information Retrieval*, 41(3):457–473, 2005.

L. S. Larkey, L. Ballesteros, and M. E. Connell. Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th*

Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02), pages 275–282, Tampere, Finland, 2002. ACM Press. ISBN 1-58113-561-0.

L. S. Larkey, L. Ballesteros, and M. E. Connell. *Light Stemming for Arabic Information Retrieval*, volume 38 of *Text, Speech and Language Technology*, pages 221–243. Springer Netherlands, 2007. ISBN 978-1-4020-6045-8.

Y.-S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. Language model based Arabic word segmentation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL '03)*, pages 399–406, Sapporo, Japan, 7–12 July 2003. Association for Computational Linguistics.

X. Liu and W. B. Croft. Statistical language modeling for information retrieval. In *the Annual Review of Information Science and Technology*, volume 39, pages 3–28. Information Today, inc., 2005.
URL: http://ciir.cs.umass.edu/pubfiles/ir-318.pdf.

J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(4):22–31, 1986.

M. Maamouri, A. Bies, H. Jin, and T. Buckwalter. Arabic Treebank: Part 1 v 2.0, 2003. LDC Catalog No. LDC2003T06.

W. Magdy, K. Darwish, O. Emam, and H. Hassan. Arabic cross-document person name normalization. In V. Cavalli-Sforza and I. Zitouni, editors, *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 25–32, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
URL: http://www.aclweb.org/anthology/W/W07/W07-0804.

D. Melamed. Automatic evaluation and uniform filter cascades for inducing N-best translation lexicons. In D. Yarovsky and K. Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 184–198, Somerset, New Jersey, 1995. Association for Computational Linguistics.

Microsoft Corporation. Arabic proofing tools in Office 2003, 2002.
URL: http://www.microsoft.com/middleeast/arabicdev/office/office2003/Proofing.asp.

Miniwatts International. Internet usage statistics — The big picture, 2007.
URL: http://www.internetworldstats.com, [Updated 30 November 2007].

H. Moukdad. Stemming and root-based approaches to the retrieval of Arabic documents on the Web. *Webology*, 3(1), March 2006.
URL: http://www.webology.ir/2006/v3n1/a22.html.

S. H. Mustafa and Q. A. Al-Radaideh. Using n-grams for Arabic text searching. *Journal of the American Society for Information Science and Technology*, 55(11):1002–1007, 2004. ISSN 1532-2882.

A. Narayanan and L. Hashem. Finite-state abstractions on Arabic morphology. *Artificial Intelligence Review*, 7(6):373–399, 1993.

D. W. Oard and F. C. Gey. The TREC 2002 Arabic/English CLIR track. In *NIST Special Publication 500-251: Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, pages 16–25, Gaithersburg, Maryland, 2002.
URL: http://trec.nist.gov/pubs/trec11/papers/OVERVIEW.gey.ps.gz.

V. M. Orengo and C. Huyck. A Stemming Algorithm for Portuguese Language. In *Proceedings of Eigth Symposium on String Processing and Information Retrieval (SPIRE '01)*, pages 186–193, 2001.

N. Ostler. *Empires of the word: A lnaguage history of the world.* Harper-Collins, New York, NY, 2005. ISBN 0-06-6621086-0.

W. Otterspeer, editor. *Studies in the History of Leiden University*, volume 5. Brill Academic Publishers, 1997. ISBN 9004090223.

P. Over, T. Ianeva, W. Kraaij, and A. Smeaton. TRECVID 2005: An overview. In *Proceedings of the TRECVID 2005 Workshop*, Gaithersburg, Maryland, 14–15 November 2006.
URL: http://www-nlpir.nist.gov/projects/tvpubs/tv5.papers/RMIT.pdf.

C. D. Paice. Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, 47(8):632–649, 1996.

W. Paik, E. D. Liddy, E. Yu, and M. McKenna. Interpretation of proper nouns for information retrieval. In *Proceedings of the workshop on Human Language Technology*, pages 309–313,

Princeton, New Jersey, 1993. Association for Computational Linguistics. ISBN 1-55860-324-7.

U. Pfeifer, T. Poersch, and N. Fuhr. Searching proper names in databases. In R. Kuhlen and M. Rittberger, editors, *Proceedings of the Hypertext - Information Retrieval - Multimedia, Synergieeffekte elektronischer Informationssysteme (HIM '95)*, volume 20 of *Schriften zur Informationswissenschaft*, pages 259–275, Konstanz, April 1995. Universitätsverlag Konstanz.

U. Pfeifer, T. Poersch, and N. Fuhr. Retrieval effectiveness of proper name search methods. *Information Processing & Management*, 32(6):667–679, 1996.

A. Pirkola, H. Keskustalo, E. Leppänen, A.-P. Känsälä, and K. Järvelin. Targeted S-gram matching: a novel n-gram matching technique for cross- and mono-lingual word form variants. *Information Research*, 7(2), 2002.
URL: http://informationr.net/ir/7-2/paper126.html.

J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 275–281, Melbourne, Australia, 1998. ACM Press.

M. Popovič and P. Willett. The effectiveness of stemming for natural-language access to Slovene textual data. *Journal of the American Society for Information Science*, 43(5): 384–390, 1992.

M. F. Porter. An algorithm for suffix stripping. *Program*, 13(3):130–137, 1980.

H. Raghavan and J. Allan. Using soundex codes for indexing names in ASR documents. In B. Ramabhadran and D. Oard, editors, *Proceedings of the HLT-NAACL 2004 Workshop: Interdisciplinary Approaches to Speech Indexing and Retrieval*, pages 22–27, Boston, Massachusetts, 6 May 2004. Association for Computational Linguistics.
URL: http://acl.ldc.upenn.edu/W/W04/W04-2905.bib.

H. Raghavan and J. Allan. Matching inconsistently spelled names in automatic speech recognizer output for information retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*

HLT/EMNLP 2005, pages 451–458, Vancouver, BC, Canada, 6–8 October 2005. The Association for Computational Linguistics.

V. Raghavan, P. Bollmann, and G. S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems*, 7(3):205–229, 1989. ISSN 1046-8188.

R. Rickman and P. Rosin. Content-based image retrieval using colour n-grams. *IEE Colloquium on Intelligent Image Databases*, pages 15/1–15/6, 22 May 1996.

A. M. Robertson and P. Willett. Applications of n-grams in textual information systems. *Journal of Documentation*, 54(1):48–67, 1998. ISSN 0022-0418.

S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

S. E. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of the Text Retrieval Conference (TREC)*, pages 151–162, Gaithersburg, Maryland, 1999. National Institute of Standards and Technology.

J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System - Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, Englewood, Cliffs, New Jersey, 1971.

R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88:1270–1278, 2000.

G. Ruibin and C. K. Yun. *An Adaptive Model for Phonetic String Search*, volume 3683 of *Lecture Notes in Computer Science*, pages 915–921. Springer Berlin, Heidelberg, 2005. ISBN 978-3-540-28896-1.

G. Salton. Computer based text retrieval. In A. Kent and J. G. Williams, editors, *Encyclopedia of Microcomputers*, volume 3, pages 82–92. Marcel Dekker, Inc., 1998.

G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989. ISBN 0-201-12227-8.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of American Society for Information Sciences*, 41:288–297, 1990.

G. Salton and M. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.

M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*, pages 33–40, Sheffield, UK, 2004. ACM Press. ISBN 1-58113-881-4.

M. Sanderson and J. Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, pages 162–169, Salvador, Brazil, 2005. ACM Press. ISBN 1-59593-034-5.

J. Savoy. A stemming procedure and stopword list for general French corpora. *Journal of the American Society for Information Science*, 50(10):944–952, 1999.

K. Shaalan and H. Raza. Person name entity recognition for Arabic. In V. Cavalli-Sforza and I. Zitouni, editors, *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 17–24, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
URL: http://www.aclweb.org/anthology/W/W07/W07-0803.

M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM '07)*, pages 623–632, Lisbon, Portugal, 6–9 November 2007. ACM Press. ISBN 978-1-59593-803-9.

K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. *Information Processing & Management*, 36(6):779–840, 2000.

A. Spink. A user-centered approach to evaluating human interaction with web search engines: an exploratory study. *Information Processing & Management*, 38(3):401–426, 2002. ISSN 0306-4573.

A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic. Searching the Web: The public and their queries. *Journal of the American Society for Information Science and Technology*, 52(3):226–234, 2001.

B. G. Stalls and K. Knight. Translating names and technical terms in Arabic text. In *Proceedings of COLING/ACL Workshop on Computational Approaches to Semitic Languages*, pages 34–41, Montreal, Quebc, Canada, 1998.

G. A. Stephen. String search. Technical Report TR-92-gas-01, School of Electronic Engineering Science, University College of North Wales, 1992.

K. Taghva, R. Elkhoury, and J. Coombs. Arabic stemming without a root dictionary. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I*, pages 152–157, Washington, DC, 2005. IEEE Computer Society. ISBN 0-7695-2315-3.

M. Tayli and A. I. Al-Salamah. Building bilingual microcomputer systems. *Communications of the ACM*, 33(5):495–504, 1990. ISSN 0001-0782.

P. Thompson and C. C. Dozier. Name searching and information retrieval. In C. Cardie and R. Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 134–140. Association for Computational Linguistics, Somerset, New Jersey, 1997.

H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '90)*, pages 1–24, Brussels, Belgium, 1990. ACM Press. ISBN 0-89791-408-2.

H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991. ISSN 1046-8188.

E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992. ISSN 0304-3975.

C. J. van Rijsbergen. *Information Retrieval*. London, 1975.
URL: http://www.dcs.gla.ac.uk/Keith/Preface.html.

P. Vines and J. Zobel. Efficient building and querying of Asian language document databases. In *Proceedings of the Fourth International Workshop on Information Retrieval for Asian Languages*, pages 118–125. Academia Sinica, Taiwan, November 1999.

P. Virga and S. Khudanpur. Transliteration of proper names in cross-language applications. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*, pages 365–366, Toronto, Canada, 2003. ACM Press. ISBN 1-58113-646-3.

T. Volkmer and S. Tahaghoghi. RMIT university shot boundary detection at TRECVID 2005. In *Proceedings of the TREC Video Retrieval Evaluation (TRECVID) Workshop*, Gaithersburg, Maryland, 14–15 November 2005.

E. M. Voorhees. Overview of TREC 2001. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 1–15, Gaithersburg, Maryland, 13–16 November 2001.

E. M. Voorhees. Overview of the TREC 2003 question answering track. In *Proceedings of the Text Retrieval Conference (TREC)*, pages 54–68, Gaithersburg, Maryland, 2003.

E. M. Voorhees and D. Harman. Overview of the nineth text retrieval conference (TREC-9). In *Proceedings of the Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, 2000.

E. M. Voorhees and D. K. Harman. Overview of the sixth TREC conference (TREC-6). In *Proceedings of the Text Retrieval Conference (TREC)*, pages 1–24, Gaithersburg, Maryland, 1997. National Institute of Standards and Technology.

R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974. ISSN 0004-5411.

S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones, and K. S. Jones. Okapi at TREC 6 automatic ad hoc, VLC, routing, filtering and QSDR. In *Proceedings of the Text Retrieval Conference (TREC)*, pages 125–136, Gaithersburg, Maryland, 1997.
URL: http://trec.nist.gov/pubs/trec6/papers/city_proc_auto.ps.

D. L. Wheeler. The Internet in the Arab world: Digital divides and cultural connections, 16 June 2004.
URL: http://www.riifs.org/guest/lecture_text/Internet_n_arabworld_all_txt.htm.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945.
URL: http://www.jstor.org/view/00994987/di009195/00p00351/0.

W. E. Winkler. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods*, pages 354–359. American Statistical Association, 1990.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc., second edition, 1999. ISBN 1-55860-570-3.

W. Wright. *A Grammar of the Arabic language*, volume 1. Librairie du Liban, Lebanon, 1874. third edition.

J. Xu, A. Fraser, and R. Weischedel. Empirical studies in strategies for Arabic retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 269–274, Tampere, Finland, 2002. ACM Press. ISBN 1-58113-561-0.

A. B. Yagoub. *Mausooat Annaho wa Assarf*. Dar Alilm Lilmalayn, Beirut, Lebanon, 1988. Third reprint, in Arabic.

J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 307–314, Melbourne, Australia, 1998. ACM Press. ISBN 1-58113-015-5.

J. Zobel and P. Dart. Finding approximate matches in large lexicons. *Software - Practice and Experience*, 25(3):331–345, 1995. ISSN 0038-0644.

J. Zobel and P. Dart. Phonetic string matching: lessons from information retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 166–172, Zurich, Switzerland, 1996. ACM Press. ISBN 0-89791-792-8.

J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38 (2):1–56, 2006. ISSN 0360-0300.

J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, 23(4):453–490, 1998.