COLING 2004

WORKSHOP ON

Computational Approaches to Arabic Script-based Languages

University of Geneva
Geneva, Switzerland
August 28, 2004

WORKSHOP THEME

Recently, there has been a surge of interest in the study of the languages of the Middle East, especially Arabic, Persian (Farsi), Pashto, Kurdish and Urdu. This sudden and urgent interest is manifested by the availability of funding for rapid development of practical systems for processing large volumes of data in these languages. Computational applications for proper name identification, entity recognition, categorization, information retrieval, summarization, machine translation and other implementations are currently in high demand. This comes at a time when advances in formal and computational linguistics over the last fifty years are being consolidated, while work on machine learning and statistical methods has been showing great promise.

There exists a considerable body of work in computational linguistics specifically targeted to these middle eastern languages. Much of the research and development has been the result of initiatives by individual research establishments or industry firms. Furthermore, the usage of the Arabic script gives rise to certain issues that are common to all these languages despite their being of distinct language families. Hence, these languages share properties such as the absence of capitalization, right to left direction, lack of clear word boundaries, complex word structure, a high degree of ambiguity due to non-representation of short vowels in the writing system, and related encoding issues.

The goal of this workshop is to provide a forum for those involved in the development of NLP systems in Arabic script languages to exchange ideas, approaches and implementations of computational systems; to discuss the common challenges faced by all practitioners; and to assess the state of the art in the field. In addition, one of the aims of the workshop is to identify promising areas for future collaborative research in the development of NLP systems for Arabic script languages. Solutions that are designed to solve the specific problems of these languages could very well have wider applications and relevance to the rest of the NLP community.

INVITED SPEAKER

Martin Kay, Stanford University

ORGANIZING COMMITTEE

Ali Farghaly, SYSTRAN Software, Inc.
Karine Megerdoomian, Inxight Software, Inc. and University of California, San Diego

PROGRAM COMMITTEE

Jan W. Amtrup, Bowne Global Solutions
Tim Buckwalter, Linguistic Data Consortium
Miriam Butt, Konstanz University, Germany
Violetta Cavalli-Sforza, Carnegie Mellon University
Joseph Dichy, Lyon University
Abdelkadir Fassi Fehri, Mohammed V University-Souissi Rabat, Morocco
Andrew Freeman, University of Washington
Nizar Habash, University of Maryland, College Park
Masayo Iida, Inxight Software, Inc
Simin Karimi, University of Arizona
Martin Kay, Stanford University
Kevin Knight, USC/Information Sciences Institute
Farhad Oroumchian, University of Wollongong in Dubai
Ahmed Rafea, The American University in Cairo
Jean Senellart, SYSTRAN Software
Bonnie Glover Stalls, University of Southern California
Rémi Zajac, SYSTRAN Software

FURTHER INFORMATIONS

Emails:

Ali Farghaly, AliFarghaly@aol.com
Karine Megerdoomian, karinem@inxight.com

www: http://members.cox.net/karinem/COLING2004

WORKSHOP PROGRAM

OPENING AND OVERVIEW

8:30 – 9:00 *Computer Processing of Arabic Script-based Languages: Current State and Future Directions*
Ali Farghaly

SESSION 1: LEXICON AND CORPORA

9:00 – 9:30 *Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools*
Mohamed Maamouri and Ann Bies

9:30 – 10:00 *Preliminary Lexical Framework for English-Arabic Semantic Resource Construction*
Anne R. Diekema

10:00 – 10:30 *The Architecture of a Standard Arabic Lexical Database: Some Figures, Ratios and Categories from the DIINAR.1 Source Program*
Ramzi Abbès, Joseph Dichy and Mohamed Hassoun

10:30 – 10:45 BREAK

SESSION 2: MORPHOLOGY

10:45 – 11:15 *Systematic Verb Stem Generation for Arabic*
Jim Yaghi and Sane Yagi

11:15 – 11:45 *Issues in Arabic Orthography and Morphology Analysis*
Tim Buckwalter

11:45 – 12:15 *Finite-State Morphological Analysis of Persian*
Karine Megerdoomian

12:15 – 2:00 LUNCH & DEMO SESSIONS

DEMONSTRATIONS

Urdu Localization Project
Sarmad Hussain

FarsiSum – A Persian Text Summarizer
Martin Hassel and Nima Mazdak

Stemming the Qur'an
Naglaa Thabet

Language Weaver Arabic->English MT
Daniel Marcu, Alex Fraser, William Wong and Kevin Knight

INVITED SPEAKER

2:00 – 2:45 *Arabic Script-Based Languages Deserve to be Studied Linguistically*
Martin Kay

SESSION 3: STATISTICAL APPROACHES

2:45 – 3:15 *An Unsupervised Approach for Bootstrapping Arabic Sense Tagging*
Mona T. Diab

3:15 – 3:45 *Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm*
Mohamed El Kourdi, Amine Bensaid and Tajje-eddine Rachidi

3:45 – 4:00 BREAK

SESSION 4: SPEECH PROCESSING

4:00 – 4:30 *A Transcription Scheme for Languages Employing the Arabic Script Motivated by Speech Processing Applications*
Shadi Ganjavi, Panayiotis G. Georgiou and Shrikanth Narayanan

4:30 – 5:00 *Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition*
Dimitra Vergyri and Katrin Kirchhoff

5:00 – 5:30 *Letter-to-Sound Conversion for Urdu Text-to-Speech System*
Sarmad Hussain

5:30 – 6:00 Discussion and Closing
Ali Farghaly and Karine Megerdoomian

Contents

Papers

Demonstrations

Computer Processing of Arabic Script-based Languages: Current State and Future Directions

Ali Farghaly
SYSTRAN Software, Inc.
9333 Genesee Ave
San Diego, CA 92121, USA
alifarghaly@aol.com

Arabic script-based languages do not belong to a single language family, and therefore exhibit different linguistic properties. To name just a few: Arabic is primarily a VSO language whereas Farsi is an SVO and Urdu is an SOV language. Both Farsi and Urdu have light verbs whereas Arabic does not. Urdu and Arabic have grammatical gender while Farsi does not. There are, however, linguistic and non-linguistic factors that bring these languages together. On the linguistic side it is the use of the Arabic script, the right to left direction, the absence of characters representing short vowels and the complex word structure. Non-linguistic common properties that bind the majority of speakers of these languages include: the Qur'an that every Moslem has to recite in Arabic, proximity of the countries speaking these languages, common history and, to a large extent, a common culture and historical influx . It is not surprising, then, that the surge of interest in the study of these languages and the sudden availability for funding to support the development of computational applications to process data in these languages come for all these languages at the same time.

This also occurs at crucial period in the field of Natural Language Processing (NLP). It is becoming increasingly evident that statistical and corpus-based approaches, though necessary, are not sufficient to address all issues involved in building viable applications in NLP. Arabic script-based languages share in different degrees an explosion of homograph and word sense ambiguity. The absence of the representation of short vowels in normal texts dramatically increases the number of ambiguities. At SYSTRAN, the average number of ambiguities of a token in many languages is 2.3, whereas in Modern Standard Arabic, it reaches 19.2. Dealing with such a problem represents a real challenge to NLP systems. Resolving ambiguity in NLP requires representation not only of linguistic and contextual knowledge but also of domain and world knowledge. It is not clear how number crunching of linguistic data could address this problem. Ambiguity in Arabic is enormous at every level: lexical, morphological and syntactic. Another serious problem is tokenization. It is extremely common in Arabic to find a token such as **"ورأيتهـــم"** which is actually a sentence consisting of a conjunction, a verb, a subject, an object in that order. Moreover, within the verb itself, there is tense, number and gender and mood. Within the object, which is only two alphabet letters, there is number, gender and case. The complexity of tokens and the abstractness of information, such as the meanings of prosodic templates (McCarthy, 1981), present challenges in the processing of Arabic script—based languages.

There has been steady progress in computational processing of Arabic script-based languages in the last few years. The greatest leap since the pioneering efforts made in the early 1980s in Arabic computational linguistics (Hlal, 1985;Ali 1985, 1987, 1988; Geith 1988; Farghaly, 1987), is the availability of Buckwalter's morphological analyzer and dictionary which has recently given a boost in that area. The great work at the LDC in the creation of a corpus of written and spoken Arabic as well as the Arabic tree bank is another important resource to the practitioners in the field. What is urgently needed in future research is work on syntactic analysis and ambiguity resolution.

Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools

Mohamed MAAMOURI
LDC, University of Pennsylvania
3600 Market Street, Suite 810
Philadelphia, PA 19104, USA
maamouri@ldc.upenn.edu

Ann BIES
LDC, University of Pennsylvania
3600 Market Street, Suite 810
Philadelphia, PA 19104, USA
bies@ldc.upenn.edu

Abstract

In this paper we address the following questions from our experience of the last two and a half years in developing a large-scale corpus of Arabic text annotated for morphological information, part-of-speech, English gloss, and syntactic structure: (a) How did we 'leapfrog' through the stumbling blocks of both methodology and training in setting up the Penn Arabic Treebank (ATB) annotation? (b) How did we reconcile the Penn Treebank annotation principles and practices with the Modern Standard Arabic (MSA) traditional and more recent grammatical concepts? (c) What are the current issues and nagging problems? (d) What has been achieved and what are our future expectations?

1 Introduction

Treebanks are language resources that provide annotations of natural languages at various levels of structure: at the word level, the phrase level, and the sentence level. Treebanks have become crucially important for the development of data-driven approaches to natural language processing (NLP), human language technologies, automatic content extraction (topic extraction and/or grammar extraction), cross-lingual information retrieval, information detection, and other forms of linguistic research in general.

The Penn Arabic Treebank began in the fall of 2001 and has now completed two full releases of data: (1) Arabic Treebank: Part 1 v 2.0, LDC Catalog No. LDC2003T06, roughly 166K words of written Modern Standard Arabic newswire from the Agence France Presse corpus; and (2) Arabic Treebank: Part 2 v 2.0, LDC Catalog No. LDC2004T02, roughly 144K words from Al-Hayat distributed by Ummah Arabic News Text. New features of annotation in the UMAAH (UMmah Arabic Al-Hayat) corpus include complete vocalization (including case endings), lemma IDs, and more specific part-of-speech tags for verbs and particles. Arabic Treebank: Part 3 is currently underway, and consists of text from An-Nahar. (Maamouri and Cieri, 2002)

The ATB corpora are annotated for morphological information, part-of-speech, English gloss (all in the "part-of-speech" phase of annotation), and for syntactic structure (Treebank II style). (Marcus, et al., 1993), (Marcus, et al., 1994)

In addition to the usual issues involved with the complex annotation of data, we have come to terms with a number of issues that are specific to a highly inflected language with a rich history of traditional grammar.

2 Issues of methodology and training with Modern Standard Arabic

2.1 Defining the specificities of 'Modern Standard Arabic'

Modern Standard Arabic (MSA), the natural language under investigation, is not natively spoken by Arabs, who acquire it only through formal schooling. MSA is the only form of written communication in the whole of the Arab world. Thus, there exists a living writing and reading community of MSA. However, the level of MSA acquisition by its members is far from being homogeneous, and their linguistic knowledge, even at the highest levels of education, very unequal. This problem is going to have its impact on our corpus annotation training, routine, and results. As in other Semitic languages, inflection in MSA is mostly carried by case endings, which are represented by vocalic diacritics appended in word-final position. One must specify here that the MSA material form used in the corpus data we use consists of a graphic representation in which short vowel markers and other pertinent signs like the 'shaddah' (consonantal germination) are left out, as is typical in most written Arabic, especially news writing. However, this deficient graphic representation does not indicate a deficient language system. The reader reads the text and interprets its meaning by 'virtually providing' the missing grammatical information that leads to its acceptable interpretation.

2.2 How important is the missing information?

Our description and analysis of MSA linguistic structures is first done in terms of individual words and then expanded to syntactic functions. Each corpus token is labeled in terms of its category and also in terms of its functions. It is marked morphologically and syntactically, and other relevant relationship features also intervene such as concord, agreement and adjacency. This redundancy decreases the importance of the absence of most vocalic features.

2.3 The issue of vocalization

The corpus for our annotation in the ATB requires that annotators complement the data by mentally supplying morphological information before choosing the automatic analysis, which amounts to a pre-requisite 'manual/human' intervention and which takes effect even before the annotation process begins. Since no automatic vocalization of unvocalized MSA newswire data is provided prior to annotation, vocalization becomes the responsibility of annotators at both layers of annotation. The part-of-speech (POS) annotators provide a first interpretation of the text/data and a vocalized output is created for the syntactic treebank (TB) annotators, who then engage in the responsibility of either validating the interpretation under their scrutiny or challenging it and providing another interpretation. This can have drastic consequences as in the case of the so-called 'Arabic deverbals' where the same bare graphemic structure can be two nouns in an 'idhafa (annexation or construct state) situation' with a genitive case ending on the second noun or a 'virtual' verb or verbal function with a noun complement in the accusative to indicate a direct object. In Example 1, genitive case is assigned under the noun interpretation, while accusative case is assigned by the same graphemic form of the word in its more verbal function (Badawi, et al., 2004, cf. Section 2.10, pp. 237-246).

Example 1[1]

Neutral form: <xbArh Al+nb> إخباره النبا

Idhafa: <ixbAruhu Al+naba>i إخبارُهُ النَبَأِ

his receipt (of) the news [news genitive]

Verbal: <ixbAruhu Al+naba>a إخبارُهُ النَبَأَ

his telling the news [news accusative]

These are sometimes difficult decisions to make, and annotators' agreement in this case is always at its lowest. Vocalization decisions have a non-trivial impact on the overall annotation routine in terms of both accuracy and speed.

Vocalization is a difficult problem, and we did not have the tools to address it when the project began. We originally decided to treat our first corpus, AFP, by having annotators supply word-internal lexical identity vocalization only, because that is how people normally read Arabic – taking the normal risks taken by all readers, with the assumption that any interpretation of the case or mood chosen would be acceptable as the interpretation of an educated native speaker annotator. In our second corpus, UMAAH, we decided that it would improve annotation and the overall usefulness of the corpus to vocalize the texts, by putting the necessary rules of syntax and vocalization at the POS level of annotation – our annotators added case endings to nouns and voice to verbs, in addition to the word-internal lexical identity vocalization. For our third corpus, ANNAHAR (currently in production), we have decided to fully vocalize the text, adding the final missing piece, mood endings for verbs. In conclusion, vocalization is a nagging but necessary "nuisance" because while its presence just enhances the linguistic analysis of the targeted corpus, its absence could be turned into an issue of quality of annotation and of grammatical credibility among Arab and non-Arab users.

3 Reconciling Treebank annotation with traditional grammar concepts in Arabic

The question we had to face in the early stages of ATB was how to develop a Treebank methodology – an analysis of all the targeted syntactic structures – for MSA represented by unvocalized written text data. Since all Arabic readers – Arabs and foreigners – go through the process of virtually providing/inserting the required grammatical rules which allow them to reach an interpretation of the text and consequent understanding, and since all our recruited annotators are highly educated native Arabic speakers, we accepted going through our first corpus annotation with that premise. Our conclusion was that the two-level annotation was possible, but we noticed that because of the extra time taken hesitating about case markings at the TB level, TB annotation was more difficult and more time-consuming. This led to including all possible/potential case endings in the POS alternatives provided by the morphological analyzer. Our choice was to make the two annotation passes equal in difficulty by transferring the vocalization difficulty to the POS level. We also thought that it is better to localize that

[1] For the transliteration system of all our Arabic corpora, we use Tim Buckwalter's code, at http://www.ldc.upenn.edu/myl/morph/buckwalter.html

difficulty at the initial level of annotation and to try to find the best solution to it. So far, we are happy with that choice. We are aware of the need to have a full and correct vocalization for our ATB, and we are also aware that there will never be an existing extensive vocalized corpus – except for the Koranic text – that we could totally trust. The challenge was and still is to find annotators with a very high level of grammatical knowledge in MSA, and that is a tall order here and even in the Arab region.

So, having made the change from unvocalized text in the 'AFP Corpus' to fully vocalized text now for the 'ANNAHAR Corpus,' we still need to ask ourselves the question of what is better: (a) an annotated corpus in which the ATB end users are left with the task of providing case endings to read/understand or (b) an annotated ATB corpus displaying case endings with a higher percentage of errors due to a significantly more complex annotation task?

3.1 Training annotators, ATB annotation characteristics and speed

The two main factors which affect annotation speed in our ATB experience are both related to the specific 'stumbling blocks' of the Arabic language.

1. The first factor which affects annotation accuracy and consistency pertains to the annotators' educational background (their linguistic 'mindset') and more specifically to their knowledge – often confused and not clear – of traditional MSA grammar. Some of the important obstacles to POS training come from the confusing overlap, which exists between the morphological categories as defined for Western language description and the MSA traditional grammatical framework. The traditional Arabic framework recognizes three major morphological categories only, namely NOUN, VERB, and PARTICLE. This creates an important overlap which leads to mistakes/errors and consequent mismatches between the POS and syntactic categories. We have noticed the following problems in our POS training: (a) the difficulty that annotators have in identifying ADJECTIVES as against NOUNS in a consistent way; (b) problems with defining the boundaries of the NOUN category presenting additional difficulties coming from the fact that the NOUN includes adjectives, adverbials, and prepositions, which could be formally nouns in particular functions (e.g., from fawq فوق NOUN to fawqa فَوقَ PREP "above" and fawqu فَوقُ ADV etc.). In this case, the NOUN category then overlaps with the adverbs and prepositions of Western languages, and this is a problem for our annotators who are linguistically savvy and have an advanced knowledge of English and, most times, a third Western language. (c) Particles are very often indeterminate, and their category also overlaps with prepositions, conjunctions, negatives, etc.

2. The second factor which affects annotation accuracy and speed is the behemoth of grammatical tests. Because of the frequency of obvious weaknesses among very literate and educated native speakers in their knowledge of the rules of '<iErAb' (i.e., case ending marking), it became necessary to test the grammatical knowledge of each new potential annotator, and to continue occasional annotation testing at intervals in order to maintain consistency.

While we have been able to take care of the first factor so far, the second one seems to be a very persistent problem because of the difficulty level encountered by Arab and foreign annotators alike in reaching a consistent and agreed upon use of case-ending annotation.

4 Tools and procedures

4.1 Lexicon and morphological analyzer

The Penn Arabic Treebank uses a level of annotation more accurately described as morphological analysis than as part-of-speech tagging. The automatic Arabic morphological analysis and part-of-speech tagging was performed with the Buckwalter Arabic Morphological Analyzer, an open-source software package distributed by the Linguistic Data Consortium (LDC catalog number LDC2002L49).

The analyzer consists primarily of three Arabic-English lexicon files: prefixes (299 entries), suffixes (618 entries), and stems (82158 entries representing 38600 lemmas). The lexicons are supplemented by three morphological compatibility tables used for controlling prefix-stem combinations (1648 entries), stem-suffix combinations (1285 entries), and prefix-suffix combinations (598 entries).

The Arabic Treebank: Part 2 corpus contains 125,698 Arabic-only word tokens (prior to the separation of clitics), of which 124,740 (99.24%) were provided with an acceptable morphological analysis and POS tag by the morphological parser, and 958 (0.76%) were items that the morphological parser failed to analyze correctly.

Items with solution	124740	99.24%
Items with no solution	958	0.76%
Total	125698	100.00%

Table 1. Buckwalter lexicon coverage, UMAAH

The ANNAHAR coverage statistics after POS 1 (dated January 2004) are as follows:

The ANNAHAR Corpus contains 340,281 tokens, of which 47,246 are punctuation, numbers, and Latin strings, and 293,035 are Arabic word tokens.

Punctuation, Numbers, Latin strings	47,246
Arabic Word Tokens	293,035
TOTAL	**340,281**

Table 2. Token distribution, ANNAHAR

Of the 293,035 Arabic word tokens, 289,722 (98.87%) were provided with an accurate morphological analysis and POS tag by the Buckwalter Arabic Morphological Analyzer. 3,313 (1.13%) Arabic word tokens were judged to be incorrectly analyzed, and were flagged with a comment describing the nature of the inaccuracy. (Note that 204 of the 3,313 tokens for which no correct analysis was found were typos in the original text).

Accurately analyzed Arabic Word Tokens	289,722	98.87%
Commented Arabic Word Tokens/ items with no solution	3,313	1.13%
TOTAL	**293,035**	**100.00%**

Table 3. Lexicon coverage, ANNAHAR

COMMENTS ON ITEMS WITH NO SOLUTION		
(no comment)	1741	52.55%
MISC comment	566	17.08%
ADJ	250	7.55%
NOUN	233	7.03%
TYPO	204	6.16%
PASSIVE_FORM	110	3.32%
DIALECTAL_FORM	68	2.05%
VERB	37	1.12%
FOREIGN WORD	34	1.03%
IMPERATIVE	24	0.73%
ADV	9	0.27%
GRAMMAR_PROBLEM	9	0.27%
NOUN_SHOULD_BE_ADJ	7	0.21%
A_NAME	6	0.18%
NUMERICAL	6	0.18%
ABBREV	5	0.15%
INTERR_PARTICLE	4	0.12%
TOTAL	3313	100.00%

Table 4. Distribution of items with no solution, ANNAHAR

4.2 Parsing engine

In order to improve the speed and accuracy of the hand annotation, we automatically pre-parse the data after POS annotation and before TB annotation using Dan Bikel's parsing engine (Bikel, 2002). Automatically pre-parsing the data allows the TB annotators to concentrate on the task of correcting a given parse and providing information about syntactic function (subject, direct object, adverbial, etc.).

The parsing engine is capable of implementing a variety of generative, PCFG-style models (probabilistic context free grammar), including that of Mike Collins. As such, in English, it gets results that are as good if not slightly better than the Collins parser. Currently, this means that, for Section 00 of the WSJ of the English Penn Treebank (the development test set), the parsing engine gets a recall of 89.90 and a precision of 90.15 on sentences of length <= 40 words. The Arabic version of this parsing engine currently brackets AFP data with recall of 75.6 and precision of 77.4 on sentences of 40 words or less, and we are in the process of analyzing and improving the parser results.

4.3 Annotation procedure

Our annotation procedure is to use the automatic tools we have available to provide an initial pass through the data. Annotators then correct the automatic output.

First, Tim Buckwalter's lexicon and morphological analyzer is used to generate a candidate list of "POS tags" for each word (in the case of Arabic, these are compound tags assigned to each morphological segment for the word). The POS annotation task is to select the correct POS tag from the list of alternatives provided. Once POS is done, clitics are automatically separated based on the POS selection in order to create the segmentation necessary for treebanking. Then, the data is automatically parsed using Dan Bikel's parsing engine for Arabic. Treebank annotators correct the automatic parse and add semantic role information, empty categories and their coreference, and complete the parse. After that is done, we check for inconsistencies between the treebank and POS annotation. Many of the inconsistencies are corrected manually by annotators or automatically by script if reliably safe and possible to do so.

4.4 POS annotation quality control

Five files with a total of 853 words (and a varying number of POS choices per word) were each tagged independently by five annotators for a quality control comparison of POS annotators. Out

of the total of 853 words, 128 show some disagreement. All five annotators agreed on 85% of the words; the pairwise agreement is at least 92.2%.

For 82 out of the 128 words with some disagreement, four annotators agreed and only one disagreed. Of those, 55 are items with "no match" having been chosen from among the POS choices, due to one annotator's definition of good-enough match differing from all of the others'. The annotators have since reached agreement on which cases are truly "no match," and thus the rate of this disagreement should fall markedly in future POS files, raising the rate of overall agreement.

5 Specifications for the Penn Arabic Treebank annotation guidelines

5.1 Morphological analysis/Part-of-Speech

The guidelines for the POS annotators are relatively straightforward, since the task essentially involves choosing the correct analysis from the list of alternatives provided by the morphological analyzer and adding the correct case ending. The difficulties encountered by annotators in assigning POS and case endings are somewhat discussed above and will be reviewed by Tim Buckwalter in a separate presentation at COLING 2004.

5.2 Syntactic analysis

For the most part, our syntactic/predicate-argument annotation of newswire Arabic follows the bracketing guidelines for the Penn English Treebank where possible. (Bies, et al. 1995) Our updated Arabic Treebank Guidelines is available on-line from the Linguistic Data Consortium at: http://www.ldc.upenn.edu/Catalog/docs/LDC2004T02/

Some points where the Penn Arabic Treebank differs from the Penn English Treebank:

- Arabic subjects are analyzed as VP internal, following the verb.
- Matrix clause (S) coordination is possible and frequent.
- The function of NP objects of transitive verbs is directly shown as NP-OBJ.

We are also informed by on-going efforts to share data and reconcile annotations with the Prague Arabic Dependency Treebank (two Prague-Penn Arabic Treebanking Workshops took place in 2002 and 2003). Some points where the Penn Arabic Treebank differs from the Prague Arabic Dependency Treebank:

- Specific adverbial functions (LOC, TMP, etc.) are shown on the adverbial (PP, ADVP, clausal) modification of predicates.
- The argument/adjunct distinction within NP is shown for noun phrases and clauses.
- Empty categories (pro-drop subjects and traces of syntactic movement) are inserted.
- Apposition is distinguished from other modification of nouns only for proper names.

In spite of the considerable differences in word order between Modern Standard Arabic and English, we found that for the most part, it was relatively straightforward to adapt the guidelines for the Penn English Treebank to our Arabic Treebank. In the interest of speed in starting annotation and of using existing tools to the greatest extent possible, we chose to adapt as much as possible from the English Treebank guidelines.

There exists a long-standing, extensive, and highly valued paradigm of traditional grammar in Classical Arabic. We chose to adapt the constituency approach from the Penn English Treebank rather than keeping to a strict and difficult adherence to a traditional Arabic grammar approach for several reasons:

- Compatibility with existing treebanks, processing software and tools,
- We thought it would be easier and more efficient to teach annotators, who come trained in Arabic grammar, to use our constituency approach than to teach computational linguists an old and complex Arabic-specific syntactic terminology.

Nonetheless, it was important to adhere to an approach that did not strongly conflict with the traditional approach, in order to ease the cognitive load on our annotators, and also in order to be taken seriously by modern Arabic grammarians. Since there has been little work done on large data corpora in Arabic under any of the current syntactic theories in spite of the theoretical syntactic work being done (Mohamed, 2000), we have been working out solutions to Arabic syntax by combining the Penn Treebank constituency approach with pertinent insights from traditional grammar as well as modern theoretical syntax.

For example, we analyze the underlying basic sentence structure as verb-initial, following the traditional grammar approach. However, since the verb is actually not the first element in many sentences in the data, we adopt a topicalization structure for arguments that are fronted before the verb (as in Example 2, where the subject is fronted) and allow adverbials and conjunctions to appear freely before the verb (as in Example 3, where a prepositional phrase is pre-verbal).

Example 2

```
(S (NP-TPC-1 Huquwq+u حُقُوقُ
         (NP Al+<inosAn+i الإنْسَان ))
   (VP ta+qaE+u  تَقَعُ
      (NP-SBJ-1 *T*)
      (PP Dimona ضِمْنَ
         (NP <ihotimAm+i+nA إهْتِمامِنا )
       )))
```

حُقُوقُ الإنْسَان تَقَعُ ضِمْنَ إهْتِمامِنا
human rights exist within our concern

Example 3

```
(S (PP min مِن
      (NP jih+ap+K جِهَةٍ
          >uxoraY أُخرَى ))
   (VP ka$af+at  كَشَفَت
      (NP-SBJ maSAdir+u  مَصادِرُ
              miSoriy~+ap+N  مِصْرِيَّةٌ
              muT~aliE+ap+N مُطَّلِعَةٌ ))
      (NP-OBJ Haqiyqata حَقِيقَةَ
             (NP Al->amri الأمر )))
```

مِن جِهَةٍ أُخرَى كَشَفَت مَصادِرُ مِصْرِيَّةٌ مُطَّلِعَةٌ حَقِيقَةَ الأمرِ
from another side, well-informed Egyptian sources revealed the truth of the matter

For many structures, the traditional approach and the treebank approach come together very easily. The traditional "equational sentence," for example, is a sentence that consists of a subject and a predicate without an overt verb (*kAna or "to be"* does not appear overtly in the present tense). This is quite satisfactorily represented in the same way that small clauses are shown in the Penn English Treebank, as in Example 4, since traditional grammar does not have a verb here, and we do not want to commit to the location of any potential verb phrase in these sentences.

Example 4

```
(S (NP-SBJ Al-mas>alatu المَسألَةُ )
   (ADJP-PRD basiyTatuN بَسِيطَةٌ))
```

المَسألَةُ بَسِيطَةٌ
the question is simple

5.3 Current issues and nagging problems

In a number of structures, however, the traditional grammar view does not line up immediately with the structural view that is necessary for annotation. Often these are structures that are known to be problematic in a more general sense for either traditional grammar or theoretical syntax, or both. We take both views into account and reconcile them in the best way that we can.

5.3.1 Clitics

The prevalence of cliticization in Arabic sentences of determiners, prepositions, conjunctions, and pronouns led to a necessary difference in tokenization between the POS files and the TB files. Such cliticized constituents are written together with their host constituents in the text (e.g., Al+<inosAn+i الإنْسَان "the person" and بِقِراءَةِ bi+qirA'ati "with reading"). Clitics that play a role in the syntactic structure are split off into separate tokens (e.g., object pronouns cliticized to verbs, subject pronouns cliticized to complementizers, cliticized prepositions, etc.), so that their syntactic roles can be annotated in the tree. Clitics that do not affect the structure are not separated (e.g., determiners). Since the word boundaries necessary to separate the clitics are taken from the POS tags, and since it is not possible to show the syntactic structure unless the clitics are separated, correct POS tagging is extremely important in order to be able to properly separate clitics prior to the syntactic annotation.

In the example below, both the conjunction *wa* "and" and the direct object *hA* "it/them/her" are cliticized to the verb and also serve syntactic functions independent of the verb (sentential coordination and direct object).

Example 5

وستشاهدونها
wasatu$AhiduwnahA
wa/CONJ+sa/FUT+tu/IV2MP+$Ahid/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I+hA/IVSUFF_DO:3FS
and + will + you [masc.pl.] + watch/observe/witness + it/them/her

The rest of the verbal inflections are also regarded as clitics in traditional grammar terms. However, for our purposes they do not require independent segmentation as they do not serve independent syntactic functions. The subject inflection, for example, appears readily with full noun phrase subject in the sentence as well (although in this example, the subject is pro-

dropped). The direct object pronoun clitic, in contrast, is in complementary distribution with full noun phrase direct objects. Topicalized direct objects can appear with resumptive pronouns in the post-verbal direct object position. However, resumptive pronouns in this structure should not be seen as problematic full noun phrases, as they are parasitic on the trace of movement – and in fact they are taken to be evidence of the topicalization movement, since resumptive pronouns are common in relative clauses and with other topicalizations.

Thus, we regard the cliticized object pronoun as carrying the full syntactic function of direct object. As such, we segment it as a separate token and represent it as a noun phrase constituent that is a sister to the verb (as shown in Example 6 below).

Example 6

```
(S wa-  -و
   (VP sa+tu+$Ahid+uwna-  سَتُشاهِدُون
       (NP-SBJ *)
       (NP-OBJ –hA  ها ))
```

وستشاهدونها
and you will observe her

5.3.2 Gerunds (Masdar) and participials

The question of the dual noun/verb nature of gerunds and participles in Arabic is certainly no less complex than for English or other languages. We have chosen to follow the Penn English Treebank practice to represent the more purely nominal *masdar* as noun phrases (NP) and the *masdar* that function more verbally as clauses (as S-NOM when in nominal positions). In Example 7, the *masdar* behaves like a noun in assigning genitive case.

Example 7

```
(PP bi- -بِـ
   (NP qirA’ati   قِراءَةِ
       (NP kitAbi   كِتابِ
           (NP Al-naHwi النَحو ))))
```

بِقِراءَةِ كِتابِ النَحو
with the reading of the book of syntax
[book genitive]

In Example 8, in contrast, the *masdar* functions more verbally, in assigning accusative case.

Example 8

```
(PP bi-  -بِـ
   (S-NOM (VP qirA’ati  قِراءَةِ)
              (NP-SBJ fATimata فاطِمَةَ -)
              (NP-OBJ Al-kitAba الكِتابَ
                    ))))
```

بِقِراءَةِ فاطِمَةَ الكِتابَ
with Fatma’s reading the book
[book accusative]

This annotation scheme to allow for both the nominal and verbal functions of *masdar* is easily accepted and applied by annotators for the most part. However, there are situations where the functions and behaviors of the *masdar* are in disagreement. For example, a *masdar* can take a determiner ‘Al-‘ (the behavior of a noun) and at the same time assign accusative case (the behavior of a verb).

Example 9

```
(PP bi  -بِـ
  (S-NOM
      (VP Al+mukal~afi  المُكَلَّف
          (NP-SBJ *)
          (NP-OBJ <injAza  إنجازَ
               (NP Al+qarAri  القَرار
                   Al+mawEuwdi
                       المَوعُودِ )))))
```

بالمُكَلَّفِ إنجازَ القَرار المَوعُودِ
with the (person in) charge of completion (of)
the promised report [completion accusative]

In this type of construction, the annotators must choose which behaviors to give precedence (accusative case assignment trumps determiners, for example). However, it also brings up the issues and problems of assigning case ending and the annotators’ knowledge of Arabic grammar and the rules of ‘<iErAb.’ These examples are complex grammatically, and finding the right answer (even in strictly traditional grammar terms) is often difficult.

This kind of ambiguity and decision-making necessarily slows annotation speed and reduces accuracy. We are continuing our discussions and investigations into the best solutions for such issues.

6 Future work

Annotation for the Arabic Treebank is on-going, currently on a corpus of An-Nahar newswire (350K words). We continue efforts to improve annotation accuracy, consistency and speed, both for POS and TB annotation.

Conclusion

In designing our annotation system for Arabic, we relied on traditional Arabic grammar, previous grammatical theories of Modern Standard Arabic and modern approaches, and especially the Penn Treebank approach to syntactic annotation, which we believe is generalizable to the development of other languages. We also benefited from the existence at LDC of a rich experience in linguistic annotation. We were innovative with respect to traditional grammar when necessary and when we were sure that other syntactic approaches accounted for the data. Our goal is for the Arabic Treebank to be of high quality and to have credibility with regards to the attitudes and respect for correctness known to be present in the Arabic world as well as with respect to the NLP and wider linguistic communities. The creation and use of efficient tools such as an automated morphological analyzer and an automated parsing engine ease and speed the annotation process. These tools helped significantly in the successful creation of a process to analyze Arabic text grammatically and allowed the ATB team to publish the first significant database of morphologically and syntactically annotated Arabic news text in the world within one year. Not only is this an important achievement for Arabic for which we are proud, but it also represents significant methodological progress in treebank annotation as our first data release was realized in significantly less time. Half a million MSA words will be treebanked by end of 2004, and our choice of MSA corpora will be diversified to be representative of the current MSA writing practices in the Arab region and the world. In spite of the above, we are fully aware of the humbling nature of the task and we fully understand and recognize that failures and errors may certainly be found in our work. The devil is in the details, and we remain committed to ironing out all mistakes. We count on the feedback of our users and readers to complete our work.

8 Acknowledgements

We gratefully acknowledge the tools and support provided to this project by Tim Buckwalter, Dan Bikel and Hubert Jin. Our sincere thanks go to all of the annotators who have contributed their invaluable time and effort to Arabic part-of-speech and treebank annotation, and more especially to our dedicated treebank annotators, Wigdan El Mekki and Tasneem Ghandour.

References

Elsaid Badawi, M. G. Carter and Adrian Gully, 2004. *Modern Written Arabic: A Comprehensive Grammar*. Routledge: New York.

Daniel M. Bikel, 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. *Proceedings of the Human Language Technology Workshop*.

Bracketing Guidelines for Treebank II Style, 1995. Eds: Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Penn Treebank Project, University of Pennsylvania, CIS Technical Report MS-CIS-95-06.

Mohamed Maamouri and Christopher Cieri, 2002. Resources for Arabic Natural Language Processing at the Linguistic Data Consortium. *Proceedings of the International Symposium on Processing of Arabic*. Faculté des Lettres, University of Manouba, Tunisia.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz & B. Schasberger, 1994. The Penn Treebank: Annotating predicate argument structure. *Proceedings of the Human Language Technology Workshop*, San Francisco.

M. Marcus, B. Santorini and M.A. Marcinkiewicz, 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics.*

Mohamed A. Mohamed, 2000. *Word Order, Agreement and Pronominalization in Standard and Palestinian Arabic*. CILT 181. John Benjamins: Philadelphia.

Zdenek Žabokrtský and Otakar Smrž, 2003. Arabic Syntactic Trees: from Constituency to Dependency. *EACL 2003 Conferenceompanion*. Association for Computational Linguistics, Hungary.

Preliminary Lexical Framework for English-Arabic Semantic Resource Construction

Anne R. Diekema
Center for Natural Language Processing
4-206 Center for Science & Technology
Syracuse, NY, 13210 USA
diekemar@syr.edu

Abstract

This paper describes preliminary work concerning the creation of a Framework to aid in lexical semantic resource construction. The Framework consists of 9 stages during which various lexical resources are collected, studied, and combined into a single combinatory lexical resource. To evaluate the general Framework it was applied to a small set of English and Arabic resources, automatically combining them into a single lexical knowledge base that can be used for query translation and disambiguation in Cross-Language Information Retrieval.

1 Introduction

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

This paper describes a preliminary framework for the creation of a lexical resource through the combination of other lexical resources. The preliminary Framework will be applied to create a translation lexicon for use in an English-Arabic CLIR system. The resulting lexicon will be used to translate English queries into (unvocalized) Arabic. It will also provide the user of the system with lexical semantic information about each of the possible translations to aid with disambiguation of the Arabic query. While the combination of lexical resources is nothing new, establishing a sound methodology for resource combination, as presented in this paper on English-Arabic semantic resource construction, is an important contribution. Once the Framework has been evaluated for English-Arabic resource construction, it can be extended to additional languages and resource types.

2 Related Work

2.1 Arabic-English dictionary combination

As pointed out previously, translation plays an important role in CLIR. Most of the CLIR systems participating in the (Arabic) Cross-Language Information Retrieval track[1] at the Text REtrieval Conference (TREC)[2] used a query translation dictionary-based approach where each source query term was looked up in the translation resource and replaced by all or a subset of the available translations to create the target query (Larkey, Ballesteros, and Connell, 2002), (Gey and Oard, 2001), (Oard and Gey, 2002). The four main sources of translation knowledge that have been applied to CLIR are ontologies, bilingual dictionaries, machine translation lexicons, and corpora.

Research shows that combining translation resources increases CLIR performance (Larkey et al., 2002) Not only does this combination increase translation coverage, it also refines translation probability calculations. Chen and Gey used a combination of dictionaries for query translation and compared retrieval performance of this dictionary combination with machine translation (Chen and Gey, 2001). The dictionaries outperformed MT. Small bilingual dictionaries were created by Larkey and Connell (2001) for place names and also inverted an Arabic-English dictionary to English-Arabic. They found that using dictionaries that have multiple senses,

[1] There have been two large scale Arabic information retrieval evaluations as part of TREC. These Arabic tracks took place in 2001, and 2002 and had approximately 10 participating teams each.

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

though not always correct, outperform bilingual term lists with only one translation alternative. Combining dictionaries is especially important when working with ambiguous languages such as Arabic.

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

2.2 Resource combination methodologies

Ruiz (2000) uses the term *lexical triangulation* to describe the process of mapping a bilingual English-Chinese lexicon into an existing WordNet-based Conceptual Interlingua by using translation evidence from multiple sources. Recall that WordNet synsets are formed by groups of terms with similar meaning (Miller, 1990). By translating each of the synonyms into Chinese, Ruiz created a frequency-ranked list of translations, and assumed that the most frequent translations were most likely to be correct. By establishing certain translation evidence thresholds, mappings of varying reliability were created. This method was later augmented with additional translation evidence from a Chinese-English parallel corpus.

A methodology to improve query translation is described by Chen (2003). The methodology is intended to improve translation through the use of NLP techniques and the combining of the document collection, available translation resources, and transliteration techniques. A basic mapping was created between the Chinese terms from the collection and the English terms in WordNet by using a simple Chinese-English lexicon. Missing terms such as Named Entities were added through the process of transliteration. By customizing the translation resources to the document collection Chen showed an improvement in retrieval performance.

3 Establishing a Preliminary Framework

The preliminary Framework provides a methodology for the automatic combination of various lexical semantic resources such as machine readable dictionaries, ontologies, encyclopedias, and machine translation lexicons. While these individual resources are all valuable individually, automatic intelligent lexical combination into one single lexical knowledge base will provide an enhancement that is larger than the sum of its parts. The resulting resource will provide better coverage, more reliable translation probability information, and additional information leveraged through the process of lexical triangulation. In an initial evaluation of the preliminary Framework, it was applied to the combination of English and Arabic lexical resources as described in section 4.

The preliminary Framework consists of 9 stages:

1) establish goals
2) collect resources
3) create resource feature matrix
4) develop evidence combination strategies and thresholds
5) construct combinatory lexical resource
6) manage problems that arise during creation
7) evaluate combinatory lexical resource
8) implement possible improvements
9) create final version of combinatory lexical resource.

Stage 1: The first stage of the Framework is intended to establish the possible usage of the combinatory lexical resource (resulting form the combination of multiple resources). The requirements of this resource will drive the second stage: resource collection.

Stage 2: Two types of resources should be collected: language processing resources such as stemmers and tokenizers; and lexical semantic resources such as dictionaries and lexicons. While not every resource may seem particularly useful at first, different resources can aid in mapping other resources together. During the second stage, conversion into a single encoding (such as UTF-8) will also take place.

Stage 3: Once a set of resources has been collected, the resource feature matrix can be created. This matrix provides an overview of the types of information found in the collected resources and of certain resource characteristics. For example, it is important to note what base form the dictionary entries have. Some dictionaries use the singular form (for nouns) or indefinite form (for verbs), some use roots, others use stems, and free resources from the web often use a combination of all of the above. By studying the feature matrix the evidence combination strategies for stage four can be developed.

	Arabic	English	word	stem	root	vocalized	unvocalized	pos	English definition	Arabic definition	synonyms	sense information
Arabeyes	x	x	x				x					
Ajeeb	x	x	x			x		x		x		x
Buckwalter	x	x		x		x	x	x	x			x
Gigaword	x		x				x					
WordNet 2.0		x						x	x		x	x

Table 1: Resource feature matrix

Stage 4: An intelligent resource combination strategy should be informed by the features of the different resources. It may be, for example, that one resource uses vocalized Arabic only and that another resource uses both vocalized and unvocalized Arabic. This fact should be taken into account by the combination strategy since the second resource can serve as an intermediary to map the first resource. Thresholding decisions are also part of stage four because the certainty of some combinations will be higher than others.

Stage 5: Stage five involves writing programs based on the findings in stage four that will automatically create the combinatory lexical resource. The combination programs should provide output concerning problematic instances that occur during the creation i.e. words that only occur in a single resource, so that these problems may be handled by alternative strategies in stage six.

Stage 6: Most of the problems in stage six are likely to be uncommon words, such as named entities or transliteration. A transliteration step, where for example English letters, i.e. *r*, are mapped to the closest Arabic sounding letters, i.e. ر , may be applied for languages that do not share the same orthographies.

Stage 7: After the initial combinatory lexical resource has been created it needs to be evaluated. First the accuracy (quality) of the combination mappings of the various resources needs to be assessed in an intrinsic evaluation. After it has been established that the combination has been successful, an extrinsic evaluation can be carried out. In this evaluation the combinatory lexical resource is tested as part of the actual application the source was intended for, i.e. CLIR. (For a more detailed description of evaluation see Section 5 below.)

Stage 8: These two evaluations will inform stage eight where possible improvements are added to the combination process.

Stage 9: The final version of the combinatory lexical resource can be created in stage nine.

4 Application of the Framework to English-Arabic

The preliminary Framework as described in section 3 was applied to five English and Arabic language resources as a kind of feasibility test. Following the Framework, we first established the goals of the combinatory lexical resource. It was determined that the resource would be used as a translation resource for CLIR that would aid query translation as well as manual translation disambiguation by the user. This meant that the combinatory lexical resource would need translation probabilities as well as English definitions for Arabic translations to enable an English language user to select the correct Arabic translation. We collected five different resources: WordNet 2.0[3], the lexicon included with the Buckwalter Stemmer[4], translations mined from Ajeeb[5], the wordlist from the Arabeyes project[6], and the LDC Arabic Gigaword corpus[7]. After the resources were collected the feature matrix was developed (see Table 1).

3 http://www.cogsci.princeton.edu/~wn

4 http://www.qamus.org

5 http://english.ajeeb.com

6 http://www.arabeyes.org

7 http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T12

The established combinatory lexical resource goals and resource feature matrix were used to determine the combination strategy. Since the resource should provide the user with definitions of Arabic words and WordNet is most comprehensive in this regard, it was selected as our base resource. The AFP newswire collection from the Gigaword corpus was used to mine Ajeeb. As is evident in the matrix, all resources contain English terms as a common denominator. The information used for evidence combination was as follows. Evidence used for mapping the Ajeeb and Buckwalter lexicons is part-of-speech information. Additionally, these two resources also provide vocalized Arabic terms/stems that can be used for a more reliable (less ambiguous) match. The Arabeyes lexicon is not terribly rich but was used as additional evidence for a certain translation through frequency weighting. The combinatory lexical resource was constructed by mapping the three lexical resources into WordNet using the evidence as discussed above (see Table 2).

world, human race, humanity, humankind, human beings, humans, mankind, man, all of the inhabitants of the earth
all of the inhabitants of the earth
عالم بشر ناس كون أرض دنيا ورى أناس آنام أنام أُناس انام برية آناس ملكوت اناس رهط المعمورة إِمْرَأ بَشَرِيّ الكائنات أنام أوادِم إِمْرُؤ بشري البشر أنَس إِنْسانِيّ إِنْس إِمْرِئ انس مَرْء راجِل مرؤ مَرْأ مَرْؤ عالَم الإنسانية مَرْئ انساني راجل إنساني امرئ مرأ الرجل مرء عِباد وَرا البشرية إمرؤ امرؤ العالم مرئ امرأ ورا رَجُل اوادم إمرئ رجل ناسُوت أنس عباد أوادم بَشَر إنس ناسوت إمرأ دُنْيا وَرَى

Table 2: Combinatory lexical resource entry example resulting from Step 5

After examining the combinatory lexical resource we found that the Arabeyes Arabic terms could not be compared directly to the Arabic terms in the other lexical resources since the determiner prefixes are still attached to the terms (as in العالم for example). More problematic were the translations mined from Ajeeb since the part-of-speech information of the Arabic term did not necessarily match the part-of-speech of the translations:

حرس#VB#2.1.2#خَفَر#خَرَسَ
#do_sentry_duty,keep_watch_over, guard,watchdog,oversee,sentinel, shield,watch,ward

The first problem is easily fixed by applying a light stemmer to the dictionary. At this point it is not clear however, how to fix the second problem. It was also decided that the translation reliability weighting by frequency is too limited to be useful. A back-translation lookup needs to determine how many other terms can result in a certain translation. This data can then update the reliability score.

5 Comprehensive Evaluation

While we only have carried out a preliminary evaluation, we envision a comprehensive evaluation in the near future. As part of this evaluation three different types of evaluation can be carried out:

1) evaluate the process of applying the Framework;
2) evaluate the combinatory lexical resource itself; and
3) evaluate the contribution of the combinatory lexical resource to the application the resource was created for.

Evaluation of the process of applying the Framework will provide evidence as to the advantages and disadvantages of our Framework, and where it may have to be adjusted.

The construction of a Combinatory Lexical Resource by applying the Framework is the first step toward an effective evaluation of the full Framework. The construction process detailed in Section 3 should be carefully documented. The evaluation will focus on the time and effort spent on the process, difficulties or ease with resources that are acquired, managed and processed, as well as problems or issues that arise during the process.

The intrinsic evaluation of the combinatory lexical resource indicates the quality of the newly created combinatory lexical resource. For this evaluation a large random number of entries will need to be evaluated for correctness. The evaluation will provide accuracy and coverage measures for the resource. Also, descriptive statistics will be generated to provide general understanding of the lexical resource that has been produced.

The extrinsic evaluation of the combinatory lexical resource is intended to measure the contribution of the resource to an application (i.e. CLIR, Information Extraction). The application of choice should be run with the combinatory lexical resource, and without. Performance metrics appropriate for the type of application can be collected for both experiments and then compared.

6 Conclusion and future research

A general Framework for lexical resource construction was presented in the context of English-Arabic semantic resource combination. The initial evaluation of the Framework looks promising in that it was successfully applied to combine five English-Arabic resources. The stages of the Framework provided a useful guideline for lexical resource combination and can be applied to resources in any language. We plan to extend the evaluation of the Framework to a more in depth intrinsic evaluation where the quality of the mappings is tested. An extrinsic evaluation should also take place to evaluate the combinatory lexical resource as part of the CLIR system. As for future research we hope to extend the evidence combination algorithms to include more sophisticated information using back translation and transliteration.

7 Acknowledgements

This work is supported by the U.S. Department of Justice.

References

A. Chen, and F. Gey. 2001. *Translation Term Weighting and Combining Translation Resources in Cross-Language Retrieval*. In "Proceedings of the Tenth Text REtrieval Conference (TREC-10)", E.M. Voorhees and D.K. Harman ed., pages 529-533, NIST, Gaithersburg, MD.

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

A. Chowdhury, M. Aljalayl, E. Jensen, S. Beitzel, D. Grossman, O. Frieder. 2002. *IIT at TREC-2002: Linear Combinations Based on Document Structure and Varied Stemming for Arabic Retrieval*. In "Proceedings of the Eleventh Text REtrieval Conference (TREC-11)", E.M. Voorhees and C.P. Buckland ed., pages 299-310, NIST, Gaithersburg, MD.

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

F.C. Gey, and Oard, D.W. 2001. *The TREC-2001 Cross-Language Information Retrieval Track: Searching Arabic using English, French, or Arabic Queries*. In "Proceedings of the Tenth Text REtrieval Conference (TREC-10)", E.M. Voorhees and D.K. Harman ed., pages 16-25, NIST, Gaithersburg, MD.

L.S. Larkey, J. Allan, M.E. Connell, A. Bolivar, and C. Wade. 2002. *UMass at TREC 2002: Cross Language and Novelty Tracks*. In "Proceedings of the Eleventh Text REtrieval Conference (TREC-11)", E.M. Voorhees and C.P. Buckland ed., pages 721-732, NIST, Gaithersburg, MD.

L.S. Larkey, L. Ballesteros, M. Connell. 2002. *Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis*. In "Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval", M. Beaulieu et al. ed., pages 275-282, ACM, NY, NY.

L.S. Larkey, and M. E. Connell. 2001. *Arabic Information Retrieval at UMass in TREC-10*. In "Proceedings of the Tenth Text REtrieval Conference (TREC-10)", E.M. Voorhees and D.K. Harman ed., pages 562-570, NIST, Gaithersburg, MD.

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

D. Oard and A. Diekema. 1998. Cross-Language Information Retrieval. *Annual Review of Information Science*, 33: 223-256.

D.W. Oard, and Gey, F.C.2002. *The TREC-2002 Arabic/English CLIR Track*. In "Proceedings of the Eleventh Text REtrieval Conference (TREC-11)", E.M. Voorhees and C.P. Buckland ed., pages 17-26, NIST, Gaithersburg, MD.

M.E. Ruiz, et al. 2001. *CINDOR TREC-9 English-Chinese Evaluation*. In "Proceedings of the 9th Text REtrieval Conference (TREC-9)", E.M. Voorhees and D.K. Harman ed., pages 379-388, NIST, Gaithersburg, MD.

J. Xu, A. Fraser, R. Weischedel. 2002. *Empirical Studies in Strategies for Arabic Retrieval*. In "Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval", M. Beaulieu et al. ed., pages 269-274, ACM, NY, NY.

The Architecture of a Standard Arabic lexical database: some figures, ratios and categories from the DIINAR.1 source program

Ramzi ABBÈS
SII[a] / SILAT[b],
ENSSIB[c],
17-21, bd. du 11 nov. 1918,
69623 Villeubanne Cedex, France
abbes@enssib.fr

Joseph DICHY
ÉLISA[d] / SILAT[b],
Université Lumière-Lyon 2,
86, rue Pasteur
69365 Lyon Cedex 07, France
joseph.dichy@univ-lyon2.fr

Mohamed HASSOUN
SII[a] / SILAT[b],
ENSSIB[c],
17-21, bd. du 11 nov. 1918,
69623 Villeubanne Cedex, France
hassoun@enssib.fr

[a] SII: *Systèmes d'Information et Interfaces*, research centre, ENSSIB

[d] ÉLISA: *Épistémologie, Linguistique, Ingénierie et Sémiologie de l'Arabe*, research centre, Lumière-Lyon 2 University

[c] ENSSIB: *École Nationale Supérieure des Sciences de l'Information et des Bibliothèque.*

[b] SILAT: *Systèmes d'information, Linguistique, Ingénierie de l'Arabe et Terminologie*, research group common to the Lyon 2 Lumière University and ENSSIB. (*Silat* is Arabic for "link", "relation".)

Abstract

This paper is a contribution to the issue – which has, in the course of the last decade, become critical – of the basic requirements and validation criteria for lexical language resources in Standard Arabic. The work is based on a critical analysis of the architecture of the DIINAR.1 lexical database, the entries of which are associated with grammar-lexis relations operating at word-form level (i.e. in morphological analysis). Investigation shows a crucial difference, in the concept of 'lexical database', between *source program* and *generated lexica*. The source program underlying DIINAR.1 is analysed, and some figures and ratios are presented. The original categorisations are, in the course of scrutiny, partly revisited. Results and ratios given here for basic entries on the one hand, and for generated lexica of inflected word-forms on the other. They aim at giving a first answer to the question of the ratios between the number of lemma-entries and inflected word-forms that can be expected to be included in, or generated by, a Standard Arabic lexical dB. These ratios can be considered as one overall language-specific criterion for the analysis, evaluation and validation of lexical dB-s in Arabic.

Keywords: Arabic lexical databases – Arabic script – word-formatives grammar – lemma-entries – morphosyntactic specifiers.

1 Introduction

In the present state of the art in the development of software and language resources in Arabic, there is an urgent need for evaluation and validation criteria based on solid analytic grounds: there exists nowadays a subsequent number of Arabic lexical databases, and more are under completion.

Existing lexical dB-s are not always, for the time being, available as such to researchers and/or developers, because they are usually embedded in software (such as a morphological analyser or a parser), and are still very difficult to make use of independently. It is to be expected, though, that the issue of availability will be overcome in a reasonably near future, and that a number of Arabic lexical databases will be found on the market, or on catalogues such as, in Europe, that of ELRA[1], and in the USA, that of LDC[2]. The on-going European project NEMLAR is presently working on the availability of language resources including lexical databases[3]. As a result, the crucial question of the quality and consistency of these databases should be met as soon as possible.

[1] European Language Resources Association, 55, rue Brillat-Savarin – 75013 Paris, France.

[2] Linguistic Data Consortium, University of Pennsylvania, 3600 Market Street, Suite 810, Philadelphia, PA 19108, USA.

[3] NEMLAR (Network for Euro-Mediterranean LAnguage Resources) is coordinated by Pr. Bente Maegaard, Center for Sprogteknologi (CST), Copenhagen. E-mail and site: nemlar@cst.dk, www.nemlar.org.

One of the criteria for the evaluation and validation of a lexical database for Arabic is both quantitative (how many?) and qualitative (what of, precisely?). In this paper, which refers to previous work on the processing of Arabic and the related lexical resources[4], we will try and give evidence on the structure of a lexical database, founded on an analysis of the DIINAR.1 database[5]. Quantitative results are only interesting if they can be interpreted in such a way as to yield information on the actual structure and categories of the lexicon of the language under consideration. We will endeavour to show that a quantitative and qualitative analysis of the lexical categories incorporated in DIINAR.1 can be interpreted with this respect. Moreover, the investigation leads to proposing a more consistent organisation of lexical information and relations, which should be included in future versions of DIINAR.

2 The type of lexical dB required by the automatic analysis of Arabic

What are the fundamental requirements of a lexical database in Arabic? The first challenge to be met upon endeavouring to build language resources in Arabic is that of the structure of the writing system of the language (Dichy, 1990), the two main features of which are: non diacriticized script in standard texts (§ 2.1) and the structure of the word-form (§ 2.2). The combined effect of these features entails the need for a lexical database that includes a subsequent number of grammar-lexis relations (§ 2.3). Such a dB is to be considered as a *sine qua non* condition of high-level and elaborate Arabic NLP.

2.1 Non diacriticized writing

It is well-known that Arabic script belongs to a group of Semitic writings originating from ancient Phoenician alphabets, such as Hebrew, Aramaic or Syriac. Phonographic translation is basically restricted to the notation of consonants and "long vowels". In the course of time, these writing systems have developed additional diacritic symbols, mainly for the needs of the oral reading of sacred texts (*Bible, New Testament, Koran*). Arabic writing has thus been provided with a sophisticated system of diacritical marks (comparable to the Massora diacritics which were later devised for the Hebrew Bible). Standard writing nevertheless disregards these symbols. This results in a high degree of homography, accounting for the multiple analyses encountered in a majority of single words by morphological analysers (which are, needless to recall, bound to consider every word off-context).

2.2 "Nucleus" and "extensions": a quick recall of the structure of word-forms in written Arabic

Unlike automatic recognition software, human readers are, of course, able to combine semantic, syntactic and morphological analyses. They are helped in their reading of Arabic written utterances by another major feature of the writing system: the very regular structure of the word-form. This structure has been introduced and extensively described previously (Desclés ed., 1983; Dichy 1984, 1990; Hassoun, 1987 – after the pioneering work of Cohen, 1961/70), and is only recalled here for the sake of clarity.

Word-forms in Arabic can be described on the whole as consisting of a *nucleus formative* (henceforth *NF*) to which *extension formatives* (henceforth *EF*) are added, either to the left or to the right (Dichy, 1997). Ante-positioned EF-s are abbreviated as *aEF*, and post-positioned ones as *pEF*. The nucleus formative, usually called *stem*, can be represented in terms of prosodic or non-concatenative morphology (after J. McCarthy's original and much discussed insights, 1981). In Semitic morphology, the stem is considered, according to a somewhat recent, but very widely followed tradition, as a compound of *root* and *pattern*. One must keep in mind, though, that many nouns cannot be analysed in such a way: they are referred to as *quasi-stems* (Dichy & Hassoun, eds., 1989).

Arabic word-forms consist of:

– *proclitics (PCL)*, which include mono-consonantal conjunctions, e.g. *wa-*, 'and' , *li-*, 'in order to', or prepositions, i.e. *bi-*, 'in, at' or 'by', etc.;
– a *prefix (PRF)*. The category, after D. Cohen's representation of the word-form, only includes

[4] The research and development work referred to in the SILAT research group goes back to the 1980ies and has been going on since (Desclés et alii 1983, Dichy & Hassoun, eds. 1989, Dichy 1984/89, 1987, 1993, 1997, 2000, Lelubre 1993, Braham 1998, Braham & Ghazali 1998). It includes a number of doctoral dissertations (Hassoun 1987, Abu Al-Chay 1988, Dichy, 1990, Gader 1992, Ghenima 1998). For further developments, see: Ezzahid 1996, Labed & Lelubre 1997, Abbas 1998, Dichy & Hassoun 1998, Ammar & Dichy 1999a et b, Abbès 1999, Dichy 1998, 2001a et b, Ghazali & Braham 2001, Lelubre 2001, Ouersighni 2002, Zaafrani 2002, Dichy & Fargali, 2003.

[5] **DIINAR.1** (*DIctionnaire INformatisé de l'Arabe*), Arabic acronym **Ma'âlî (**"Mu'jam al-'Arabiyya l-'âlî"), is a comprehensive Arabic Language dB operating at word-form level (morphological analysis or generation). It has been completed in close cooperation, in Tunis by IRSIT (now SOTETEL-IT - A. Braham and S. Ghazali), and in France by ENSSIB (M. Hassoun) and the Lumière-Lyon 2 University (J. Dichy). See Dichy, Braham, Ghazali & Hassoun, 2002.

the prefixes of the imperfective, e.g., *ya-*, prefixed morpheme of the 3rd person;
– a *stem*, which can be represented in terms of a ROOT (an ordered triple of consonants, or, by extension of the system, a quadruple) and a PATTERN (roughly: a template of syllables, the consonants of which are the triple of the ROOT to which monoconsonantal affixes are added). The stem *takabbar*, 'to be haughty', thus consists of the 3-consonant ROOT /k-b-r/ and of the PATTERN /taR^{1}aR^{2}R^{2}aR^3/, where R^1, R^2 and R^3 stand for 'radical consonant 1, 2, 3', and are instantiated by the triple of the ROOT (R^1=*k*, R^2=*b*, R^3=*r*). Nouns that cannot be analysed in ROOT and PATTERN are conventionally referred to as *quasi-stems*, e.g.: *'ismâ'îl*, 'Ishmael', *yûnîskû*, 'UNESCO', *kahramân*, 'amber';
– *suffixes (SUF)*, such as verb endings, nominal cases, the nominal feminine ending *-at*, etc.;
– *enclitics (ECL)*. In Arabic, enclitics are complement pronouns.

In the table below two apparently equivalent representations of the structure of the Arabic word-form are given. The main difference between them lies in the fact that (2) aims at highlighting the relations between nucleus and extension formatives (NF and EF-s), featuring a triangle (in boldface below). The rules governing the relations between morphemes embedded in the word-form are included in a *word-formatives grammar* (henceforth *WFG* – Dichy, 1987, 1997). These rules, and the features they involve, are distributed along these three relations, a great number of which are related to the lexical nucleus, and have to rely upon the finite set of grammar-lexis relations operating at word-level (formalised in Dichy, 1990).

(1) 'Traditional' representation of the word-form (D. Cohen, 1961/70, Desclés, ed., 1983)	maximal
	_____word-form_____
	| |
	minimal
	__word-form__
	| |
	##PCL # PRF +STEM+ SUF # ECL##
	STEM = <ROOT¤¤PATTERN>

(2) Nucleus-extensions representation (Dichy, 1997)	**NF** **/ ** **aEF — pEF** / \ / \ PCL PRF SUF ECL

Table 1: Structure of the word-form in Arabic

2.3 Word-formatives grammar (WFG) and word-level grammar-lexis relations

Complex as it may appear, the above structure is regular, and remains, up to a certain point, recognisable from a psychological stand. It is, subsequently, very restrictive: Arabic word-forms include one lexical stem and one only[6]. In fact, the word-formatives grammar (WFG) accounts for the regular structure of the word-form.

Rules involving word-formatives (the above nucleus and extension formatives, NF and EF) are based on three fundamental types of relations (Dichy, 1987): ⇒ 'entails', **≠>** 'excludes', ****** 'is compatible with' (or 'admits'), the third of which is attached to the opposed pair of the first two as an 'elsewhere' relation of a special kind, directly connected to ambiguity in language analysis processes (Dichy, 2000). In generation, all 'compatibility' (or 'admit') relations can in fact be rewritten in terms of 'entail' or 'exclude' rules associated with specific sets of word-formatives. 'Compatibility' relations are mostly useful in the formalisation of recognition rules, when ambiguity is at stake[7]. The formal structure of the WFG thus includes relations of the three types above, which are, in turn, involved in either one of the two following combination schemes:

- **EF ↔ EF combinations**, such as PCL → SUF rules, e.g.:
  PCL = *bi* ⇒ SUF = {*i, in, a, an, îna, î, ayni, ay*}
  which can be phrased as: 'the proclitic preposition *bi#* entails one of the indirect (or genitive) case suffixes'. Other rules will point to a given case suffix in a given utterance.

- **NF ↔ EF combinations**, such as STEM → SUF rules, e.g.:
  STEM = 'diptote' ⇒ SUF = {*u, a, i*}
  which can be phrased as: 'a stem whose declension is diptote entails case-endings belonging to the listed set'. (Diptote stems may also be compatible with dual or plural suffixes, which is taken into account in another rule.)

Another type of relation to be encoded in a lexical database is:

6. A few exceptive compound items exist, but they are kept marginal by the structure of the language, for the obvious reasons hinted at here, unlike what has happened in Modern Hebrew, as opposed to the Biblical and Medieval state of the language (Kirtchuk, 1997).

7. Automatic recognition and generation are not to be considered as reverse processes. Evidence from Arabic is given in Desclés, ed., 1983; Dichy, 1984, 1997, 2000.

- **NF ↔ NF linking combinations**, which have to be encoded whenever the morphological variation is not rule-predictable (cf. Mel'čuk's concept of *syntactic*, 1982). This is the case in a majority of singular ↔ 'broken plural' links in nouns or adjectives, as well as in 'perfective' ↔ 'imperfective' (*mâ**d**î* ↔ *mu**d**âri'*) links, in verbs belonging to 'simple' PATTERNS (*al-fi'l al-mujarrad*).

In an Arabic lexical dB, lexical entries (NF-s or STEMS in the above representation) need to be associated with ***morphosyntactic specifiers*** ensuring their insertion in word-forms, and their morphological variation (conjugation or declension). Morphosyntactic specifiers, in other words, account for:
– grammar-lexis relations, i.e. *NF ↔ EF combinations*;
– morpho-lexical variation, i.e. *NF ↔ NF linking combinations*.

Lexical entries thus 'entail', 'exclude' or 'admit' a number of grammatical morphemes listed in the various fields of the word-form as word-formatives, either on a non regular basis, or on the basis of rules founded on semantic features that cannot be deduced from the formal structure of the morpheme. As shown in Dichy (1997), morphosyntactic specifiers make up formally, in a lexical database, for information associated in the speaker's memory to various levels of linguistic analysis (morpho-phonological, syntactic or semantic features).

This structure has often been disregarded in the elaboration of Arabic lexical databases on the assumption that the representation of lexical entries as a mere combination of PATTERN and ROOT (plus a number of suffixes) is sufficient. This is definitely *not* the case: evidence recalled in this paragraph (also in Hassoun & Dichy, eds. 1989, Dichy, 1997, Dichy & Fergaly, 2003) show that grammar-lexis relations operating at word-form level can only be taken into account if information is associated to whole stems (or nuclei), or to stem+suffix 'frozen' compounds. These relations cannot be predicted on the sole basis of patterns.

The description of the WFG outlined in this paragraph has led to the elaboration of *exhaustive and finite sets of morphosyntactic specifiers* liable to be associated to the *non finite lexical entries* of an Arabic database (Dichy, 1997). These sets have been associated with the entries of the DIINAR.1 Arabic Language database. The WFG has been on the other hand implemented in the related generation and analysis source programs.

Another lexical LR including morphosyntactic information at word level is the lexicon elaborated and completed by Timothy Buckwalter, which has been used in the finite-state morphological analyser elaborated at the European Xerox Research Centre (Meylan, France)[8].

3 A few figures and ratios from DIINAR.1: generated lexica vs. source lexicon

In the previous section, we outlined the structure of the WFG and the information associated with lexical entries in the source program of the DIINAR.1 database.

It is essential to note that the expression *lexical database* is ambiguous, i.e. that it is liable to refer, either:
– to a *source program* drawing on lists of *basic* lexical or grammatical items (related to a grammar of the kind outlined in the previous section),
– or a set of *generated lexica*, the items of which can be either *basic* (as in the source program) or *combined*, i.e. resulting from the combination of basic items, according to the rules of the word-formatives grammar.

Software relying partly or entirely on morphological analysis may, or may not, need all the information outlined in section 2. They draw on lexica generated by the source program associated with the dB (Hassoun, 1987). Generated lexica can be restricted to a subset of information, as in a spelling checker (Gader, 1992), or extended to all available information, as in a parser (Ouersighni, 2002) or in an interactive language teaching software (Zaafrani, 2002). In the current section, we will examine the architecture of the DIINAR.1 database, from the standpoint of the relation between the figures of the basic entries included (§ 3.1 and 3.2), and that of the inflected word-forms (§ 3.3).

3.1 The basic figures of the DIINAR.1 source program

The total number of lemma-entries in the DIINAR.1 database is : 121,522.

This includes 445 tool-words belonging to various grammatical categories (e.g.: prepositions, conjunctions, etc.) and the prototype of a proper names database of 1,384 entries. Both types of entries are associated with a particular word-formatives grammar, and with their own subsets of morpho-syntactic specifiers.

The main parts of the database include:

[8]. Beesley, 1998, 2001, Beesley and Karttunen, 2003. Also: Buckwalter, 2002.

Nouns, including adjectives	29,534
[Broken plural nominal forms]	[9,565]
Verbs	19,457
Deverbals:	
- infinitive forms (*masdar*)	23,274
- active participles (*'ism al-fâ'il*)	17,904
- passive participles (*'ism al-maf'ûl*)	13,373
- 'analogous adjectives' (*sifa musabbaha*)	5,781
- 'nouns of time & place' (*'ism al-makân wa-z-zamân*)	10,370
Total number of deverbals	[70,702]
Subtotal of lemmas	119,693

Table 2: Number of lemmas and items belonging to main major lexical categories

3.2 Comments and critical analysis

(1)

Table 2 features two ratios of general interest for the structure of the Arabic general Lexicon:

– The ratio between broken plural nominal forms (which are not counted as lemmas[9]) and nouns and adjectives is roughly of one to four.
– Deverbals appear to be 3.6 more numerous than verbs.

(2)

The above categorisation follows that of traditional Arabic grammar. Two sub-categorisations should, nevertheless be revisited for linguistic consistency reasons:

– Adjectives (although they can appear as nouns in many syntactic structures) should be isolated. This will be needed, of course, in parsing – even in 'shallow parsing'. Adjectives in Arabic can be identified through syntactic tests.
– 'Nouns of time and place' (*'asmâ'^{u} l-makân wa-z-zamân*) should not, in future versions of DIINAR, remain in the 'deverbal' category. They are in fact (except for the earliest stages in the development of the language) inserted in syntactic structures as full nouns.

(3)

It is to be noted, on the other hand, that (except for 'nouns of time and place') DIINAR.1 is very consistent in distinguishing between nouns and deverbals: deverbals re-used as nouns, and showing full nominal features appear, in the dB, twice (as 'deverbals' *and* as 'nouns', with their related morphosyntactic specifiers), e.g.:

- *sâkin*, plur. *sâkinûn, sâkinât*, 'dwelling', 'inhabiting', is a deverbal, e.g.:
  *Na**h**nu sâkinûna madînat^{a} al-'iskandariyya* = 'We live in Alexandria'.
- *sâkin*, plur. *sukk**ā**n* (broken plural form), 'inhabitant', is a full noun (appearing in the first line of Table 2), e.g.:
  *Na**h**nu sukkân^{u} madînat^{i} l-'iskandariyya* = 'We are the inhabitants of Alexandria'.

(4)

The number of roots in DIINAR.1 is 6,546, it being understood that a great many nouns cannot be analysed in ROOT and PATTERN. (On the other hand, *all* the verbs and deverbals of the language can – Dichy, 1984/89.)

3.3 The DIINAR.1 lexica of inflected word-forms

The number of combined proclitics (which are effectively in use in Modern Standard Arabic), suffixes, prefixes and enclitics is shown in the tables below:

Proclitics (combined)	64
Prefixes	8
Suffixes (combined)	67
Enclitics	13

Table 3: Total number of extension formatives (EF-s)

	Associated with nouns	Associated with verbs	Common to both types
Proclitics	44	13	7
Prefixes	0	8	0
Suffixes	11	42	0
Enclitics	1	1	11

Table 4: EF-s associated with nominal and/or verbal stems

It is easy to imagine, on the basis of the above table, that one could generate huge figures through multiplying the number of extension formatives among themselves, then multiplying the result by the number of nouns and/or verbs. In order to avoid 'over-powerful' inflation of data, a consistent database needs to be filtered through (a) a word-formatives grammar and (b) morphosyntactic specifiers associated to stems.

[9]. 'Broken plural' forms are related to a singular noun-form lemma. Links between singular and plural forms, in the dB, are described as *NF ⟷ NF linking combinations* (see § 2.3).

The overall figures for inflected forms lexica generated by the DIINAR.1 can be broken down as shown in Table 5:

	a Number of nuclei or stems	**b** Number of inflected forms	**b/a** ratio
Verbs	19,457	3,060,716	157.3
Deverbals	70,702	2,909,772	41.15
Nouns and adjectives (+broken plurals)	39,099	1,781,316	45.55
Gramma-tical words	445	---	---
Proper names	1,384	11,403	8.23
Total figure and ratio	131,087	7,774,938	59.31

Table 5: Inflected word-forms, i.e., 'minimal word-forms' (see Table 1)

3.4 The fundamental ratio between lemma-entries and inflected word-forms

High as they may seem, the above figures are not over-powerful, and result from stem-by-stem filtering of information through morphosyntactic specifiers and the associated word-formatives grammar.

One can also compare the ratio between the total number of stems and that of inflected forms to what can be found in another language, which is equally known to be a highly inflected one. The Xerox Spanish Lexical Transducer contained, in 1996 over 46,000 base-forms, and generated over 3,400,000 inflected word-forms (Beesley & Karttunen, 2003, p. xvii). The ratio between inflected forms and base-forms in the Xerox Spanish database was then of around 74 to one. In the DIINAR.1 dB, the same ratio is of just under 60 to one, which can be considered as reasonable.

The question of how many 'maximal word' forms can be correctly generated remains to be introduced and discussed in a further paper.

4 The rationale beyond ratios: towards a first set of validation criteria for Arabic lexica

The ratios considered in the present paper are divided in two general categories:

- The category encountered in § 3.2 involves NF ↔ NF linking combinations (§ 2.3):
  (a) The ratio between the number of noun lemmas (in general vocabulary) and that of 'broken plurals' is of 1 'broken plural' for every 4 nouns.
  (b) The overall ratio between verbs and deverbals gives an average of 3.6 deverbals for one verb.

- The ratios given in § 3.3 and 3.4 consider the number of basic entries, such as nouns, verbs, deverbals, etc., and the inflected forms generated through the rules of the WFG and the grammar-lexis relations specifiers included in the dB. In nouns, the relatively high ratio of 45.55 is due to the combination of case-endings with other suffixes. In proper names, case-endings are limited, because they do not vary according to definiteness or indefiniteness, and also because some categories of proper names are in addition not liable to be followed by the relative suffix *–iyy*).

In this contribution, the numbers of lemma-entries reflect the state of the DIINAR.1 database, which is likely to be modified, in the course of time, through eliminating lemmas corresponding to words that have fallen out of use or through adding new entries. Ratios, on the other hand, reflect the word-formatives grammar as well as the overall structure of the sets of morpho-syntactic specifiers associated to lexical entries. They are, on he whole, to remain stable. It is therefore reasonable to consider that they should be added to the language-specific parts of a check-list devised for the evaluation and validation of Arabic lexical resources, or of multilingual lexica including Arabic.

5 Acknowledgements

The present work is presented with the support of the NEMLAR Euro-Mediterranean project (see note 3).

The generated lexica of DIINAR.1 are to be made available through the European Language Resources Association (ELRA), and The Evaluation and Language resources Distribution Agency (ELDA), Paris, http://www.elda.fr.

References

Wijdan Abbas Mekki. 1998. Définition et description des unités linguistiques intervenant dans l'indexation automatique des textes en

arabe, Doct. Dissert., ENSSIB/Université Lyon 2.

Ramzi Abbès. 1999. *Conception d'un prototype de concordancier de la langue arabe*, Mémoire de DEA en Sciences de l'information et de la communication, ENSSIB.

Najim Abu Al-Chay. 1988. *Un Système expert pour l'analyse et la production des verbes arabes dans une perspective d'Enseignement Assisté par Ordinateur*. Doct. Dissert., Université Lyon 1.

S. Ammar. & J. Dichy. 1999a. *Les verbes arabes*, Paris, Hatier (collection Bescherelle - Original introduction in French).

— 1999b. *Al-'Af'âl al-'arabiyya*, Paris, Hatier (collection Bescherelle - Original Arabic introduction).

Kenneth Beesley. 1989/91. "Computer Analysis of Arabic Morphology: A two-level approach with detours." In Bernard Comrie and Mushira Eid, eds., 1991. *Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics*, Amsterdam, John Benjamins: 155-172.

— 2001. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *ACL 39th Annual Meeting.* Workshop on *Arabic Language Processing; Status and Prospect*, Toulouse: 1-8.

Kenneth Beesley & Lauri Karttunen. 2003. *Finite State Morphology.* CSLI Publications, Stanford, California.

Abdelfattah Braham & Salem Ghazali. 1998. Qâ'idatu l-bayânât al-mu'jamiyya al-'arabiyya, 'aw ma<u>s</u>rû' Mu'jam al-'Arabiyya l-'âliyy, 'Ma'âlî-DIINAR', **h**as**î**la wa-'âfâq. *Al-Majalla l-'Arabiyya li-l-'ulûm*, 32: 14-23.

Tim Buckwalter. 2002. *Buckwalter Arabic Morphological Analyzer Version 1.0*. Linguistic Data Consortium, catalog number LDC2002L49 and ISBN 1-58563-257-0. <http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49 >

David Cohen. 1961/70. "Essai d'une analyse automatique de l'arabe". *T.A. informations*, 1961. Reprod. in D. Cohen. 1970. *Études de linguistique sémitique et arabe*. Paris, Mouton: 49-78.

Jean-Pierre Desclés, ed. 1983. (H. Abaab, J.-P. Desclés, J. Dichy, D.E. Kouloughli, M.S. Ziadah). *Conception d'un synthétiseur et d'un analyseur morphologiques de l'arabe, en vue d'une utilisation en Enseignement assisté par Ordinateur*, Rapport rédigé sous la direction de J.-P. Desclés, à la demande du Ministère français des Affaires étrangères (sous-direction de la Politique linguistique).

Joseph Dichy. 1984/89. "Vers un modèle d'analyse automatique du mot graphique non-vocalisé en arabe", in Dichy & Hassoun, 1989: 92-158.

— 1987. "The SAMIA Research Program, Year Four, Progress and Prospects". *Processing Arabic Report* 2, T.C.M.O., Nijmegen University: 1-26.

— 1990. *L'Écriture* dans *la représentation de la langue : la lettre et le mot en arabe*. State Doct. Dissert. thèse d'État (en linguistique), Université Lumière-Lyon 2.

— 1993. "Knowledge-system simulation and the computer-aided learning of Arabic verb-form synthesis and analysis". *Processing Arabic Report* 6/7, T.C.M.O., Nijmegen University: 67-84, 92-95.

— 1997. "Pour une lexicomatique de l'arabe : l'unité lexicale simple et l'inventaire fini des spécificateurs du domaine du mot". *Meta* 42, spring 1997, Québec, Presses de l'Université de Montréal: 291-306.

— 1998. "Mémoire des racines et mémoire des mots : le lexique stratifié de l'arabe". T. Baccouche, A. Clas et S. Mejri, eds., *La Mémoire des mots.* Special issue of: *Revue Tunisienne de Sciences Sociales*, 117: 93-107.

— 2000. "Morphosyntactic Specifiers to be associated to Arabic Lexical Entries - Methodological and Theoretical Aspects". Proceedings of the *ACIDA' 2000* conference, Monastir (Tunisia), 22-24 March 2000, *Corpora and Natural Language Processing* vol.: 55-60.

— 2001a. "Une première classification des verbes arabes en fonction de leur structure d'arguments". A. Fassi Fehri, ed., Actes du colloque international *Génération Systématique de la langue et Traduction automatique*, (Rabat, 15-17 novembre 1999). Special issue of: *Recherches Linguistiques*, IERA, May 2001, vol. 2 : 39-70.

— 2001b. "On lemmatization in Arabic. A formal definition of the Arabic entries of multilingual lexical databases". *ACL 39th Annual Meeting.* Workshop on *Arabic Language Processing; Status and Prospect*, Toulouse: 23-30.

J. Dichy, A. Braham, S. Ghazali, M. Hassoun. 2002. La base de connaissances linguistiques DIINAR.1 (DIctionnaire INformatisé de l'Arabe, version 1), Proceedings of the *International Symposium on The Processing of Arabic*, Tunis (La Manouba University), 18-20 April 2002.

J. Dichy & Ali Fargaly. 2003. Roots & Patterns vs. Stems plus Grammar-Lexis Specifications: on what basis should a multilingual lexical database centred on Arabic be built? *IX^th^ Machine Translation Summit* (New-Orleans, Sept. 23-27, 2003), Proceedings of the Workshop on *Machine Translation for Semitic Languages: Issues and Approaches*: 1-8

J. Dichy & M.O. Hassoun, eds. 1989. *Simulation de modèles linguistiques et Enseignement Assisté par Ordinateur de l'arabe - Travaux SAMIA I*. Paris, Conseil International de la Langue Française.

J. Dichy & M.O. Hassoun. 1998. Some aspects of the DIINAR-MBC research programme". In A. Ubaydly, ed., 1998: 2.8.1-6.

Everhard Ditters, ed. 1986-1995. *Processing Arabic Report* 1 (1986), 2 (1987), 3 (1988), 4 (1989), 5 (1990), 6/7 (1993), 9 (1995), Institute for the Languages and Cultures of the Middle-East., Nijmegen University.

Samia Ezzahid. 1996. *Méthodologie d'élaboration d'une base de données lexicale de l'arabe (vocabulaire général) d'après la théorie Sens-Texte d'Igor Mel'cuk*. Doct. diss. Université Lyon 2.

Bernard Fradin. 1994. L'approche à deux niveaux en morphologie computationnelle et les développements récents de la morphologie", in B. Fradin, ed., *Morphologie computationnelle*, *T.A.L.*, 35, 1994-2, Paris, ATALA: 9-48.

Nabil Gader. 1992. *Conception et réalisation d'un prototype de correcteur orthographique de l'arabe*. Mémoire de DEA en Sciences de l'information et de la communication, ENSSIB.

Salem Ghazali & Abdelfattah Braham. 2001. "Dictionary Definitions and Corpus-Based Evidence in Modern Standard Arabic". In *ACL 39^th^ Annual Meeting*. Workshop on *Arabic Language Processing; Status and Prospect*, Toulouse: 51-57.

Malek Ghenima. 1998. *Analyse morpho-syntaxique en vue de la voyellation assistée par ordinateur des textes écrits en arabe*. Doct. dissert., ENSSIB/Université Lyon 2.

Mohamed Hassoun. 1987. *Conception d'un dictionnaire pour le traitement automatique de l'arabe dans différents contextes d'application*. State doct. dissert., Université Lyon 1.

Pablo Kirtchuk. 1997. Renouvellement grammatical, renouvellement lexical et renouvellement conceptuel en sémitique, in C. Boisson & Ph. Thoiron, eds, 1997, *Autour de la dénomination*, Presses Universitaires de Lyon: 41-69.

Lamia Labed & Xavier Lelubre. 1997. "DIINAR-TOPT: conception d'une base de données terminologique Arabe/français dans le domaine de l'optique". In *JST'97: 1ères JST FRANCIL 1997: L'ingénierie de la langue : de la recherche au produit*, Avignon, 15-16 avril 1997, Aupelf-Uref/Francil: 523-8.

Xavier Lelubre. 1993. "Courseware for the theory and practice of Arabic conjugation". *Processing Arabic Report*, 6/7, TCMO, Nijmegen University: 85-89 and 92-95.

— 2001. "A Scientific Arabic Terms Data Base: Linguistic Approach for a Representation of Lexical and Terminological Features". In *ACL 39^th^ Annual Meeting*. Workshop on *Arabic Language Processing; Status and Prospect*, Toulouse: 66-72.

Igor Mel'cuk. 1982. *Towards a Language of Linguistics, A System of Formal Notions for Theoretical Morphology*, München : Wilhem Fink Verlag.

Riadh Ouersighni. 2002. *La conception et la réalisation d'un système d'analyse morpho-syntaxique robuste pour l'arabe : utilisation pour la détection et le diagnostic des fautes d'accord*. Doct. dissert., ENSSIB/Université Lyon 2.

SAMIA [Research Group]. 1984. "Enseignement Assisté par Ordinateur de l'arabe. Simulation à l'aide d'un modèle linguistique, la morphologie". *Actes du Colloque "E.A.O. 84"*, (Lyon, 4-5 septembre 1984), Paris, Agence de l'informatique: 81-96.

Ahmad Ubaydly, ed. 1998. *Proceedings of the 6th International Conference and Exhibition on Multilingual Computing (ICEMCO 98)*, Centre of Middle Eastern Studies, University of Cambridge.

Riadh Zaafrani. 2002. *Développement d'un environnement interactif d'apprentissage avec ordinateur de l'arabe langue étrangère*. Doct. dissert., ENSSIB/Université Lyon 2.

Systematic Verb Stem Generation for Arabic *

Jim Yaghi
DocRec Ltd.,
34 Strathaven Place, Atawhai,
Nelson, New Zealand.
jim@docrec.com

Sane M Yagi
Department of English,
University of Sharjah,
Sharjah, U.A.E.
saneyagi@yahoo.com

Abstract

Performing root-based searching, concordancing, and grammar checking in Arabic requires an efficient method for matching stems with roots and vice versa. Such mapping is complicated by the hundreds of manifestations of the same root. An algorithm based on the generation method used by native speakers is proposed here to provide a mapping from roots to stems. Verb roots are classified by the types of their radicals and the stems they generate. Roots are moulded with morphosemantic and morphosyntactic patterns to generate stems modified for tense, voice, and mode, and affixed for different subject number, gender, and person. The surface forms of applicable morphophonemic transformations are then derived using finite state machines. This paper defines what is meant by 'stem', describes a stem generation engine that the authors developed, and outlines how a generated stem database is compiled for all Arabic verbs.

1 Introduction

Morphological parsers and analysers for Arabic are required to dissect an input word and analyse its components in order to perform even the simplest of language processing tasks. The letters of the majority of Arabic words undergo transformations rendering their roots unrecognisable. Without the root, it is difficult to identify a word's morphosemantic template, which is necessary for pinpointing its meaning, or its morphosyntactic pattern, which is essential for realising properties of the verb, such as its tense, voice, and mode, and its subject's number, gender, and person. It is fundamental that an analyser be able to reverse the transformations a word undergoes in order to match the separated root and template with the untransformed ones in its database. Unfortunately, defining rules to reverse transformations is not simple.

* The authors wish to thank the anonymous reviewers of this article as their suggestions have improved it significantly.

Research in Arabic morphology has primarily focused on morphological analysis rather than stem generation.

Sliding window algorithms (El-Affendi, 1999) use an approximate string matching approach of input words against lists of roots, morphological patterns, prefixes, and suffixes. Algebraic algorithms (El-Affendi, 1991), on the other hand, assign binary values to morphological patterns and input words, then perform some simple algebraic operations to decompose a word into a stem and affixes. Permutation algorithms (Al-Shalabi and Evens, 1998) use the input word's letters to generate all possible trilateral or quadrilateral sequences without violation of the original order of the letters which is then compared with items in a dictionary of roots until a match is found. Linguistic algorithms (Thalouth and Al-Dannan, 1990; Yagi and Harous, 2003) remove letters from an input word that belong to prefixes and suffixes and place the remainder of the word into a list. The members of this list are then tested for a match with a dictionary of morphological patterns.

The primary drawback of many of these techniques is that they attempt to analyse using the information found in the letters of the input word. When roots form words, root letters are often transformed by replacement, fusion, inversion, or deletion, and their positions are lost between stem and affix letters. Most attempts use various closest match algorithms, which introduce a high level of uncertainty. In this paper, we define Arabic verb stems such that root radicals, morphological patterns, and transformations are formally specified. When stems are defined this way, input words can be mapped to correct stem definitions, ensuring that transformations match root radicals rather than estimate them.

Morphological transformation in our definition is largely built around finite state morphology (Beesley, 2001) which assumes that these transformations can be represented in terms of regular relations between regular language forms. Beesley (2001) uses finite state transducers to encode the

intersection between roots, morphological patterns, and the transformation rules that account for morphophonemic phenomena such as assimilation, deletion, epenthesis, metathesis, etc.

In this paper, a description of the database required for stem generation is presented, followed by a definition of stem generation. Then the database together with the definition are used to implement a stem generation engine. This is followed by a suggestion for optimising stem generation. Finally, a database of generated stems is compiled in a format useful to various applications that the conclusion alludes to.

In the course of this paper, roots are represented in terms of their ordered sequence of three or four radicals in a set notation, i.e., {F,M,L,Q}. When the capitalised Roman characters F, M, L, and Q are used, they represent a radical variable or place holder. They stand for First Radical (F), Medial Radical (M), Last Radical in a trilateral root (L), and Last Radical in a quadrilateral root (Q).

For readability, all Arabic script used here is followed by an orthographic transliteration between parentheses, using the Buckwalter standard[1]. Buckwalter's orthographic transliteration provides a one-to-one character mapping from Arabic to US-ASCII characters. With the exception of a few characters, this transliteration scheme attempts to match the sounds of the Roman letters to the Arabic ones. The following list is a subset of the less obvious transliterations used here: ء (@), ى (Y), ◌َ (a), ◌ِ (i), ◌ُ (u), ◌ْ (o), and ◌ّ (~).

2 Stem Generation Database

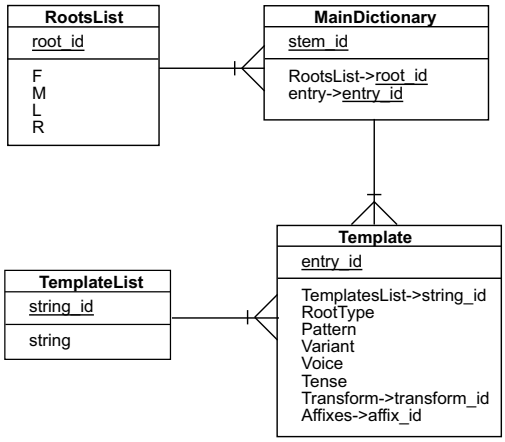


Figure 1: The stem generation database tables and their relations.

[1]The complete table of orthographic transliteration may be found at http://www.qamus.org/transliteration.htm

Arabic stems can be generated if lists of all roots and all morphological patterns are provided. It is necessary that this data be coupled with a database that links the roots with their morphological patterns (or templates) so that only valid stems are generated for each root. The roots in this database may be moulded with morphosemantic and morphosyntactic patterns to generate intermediate form stems. The stems may then be transformed into final surface forms with a number of specific morphophonemic rules using a finite state transducer compiling language.

Figure 1 shows a summary of the stem generation tables and their relations. The *RootsList* table contains all verb roots from the popular Arabic dictionary, Al-Waseet, (Mustapha et al., 1972), with F, M, L, and Q representing the table fields for up to four radicals per root. A root identifier is used to link this table to the *Template* table. The *Template* table lists all morphosemantic and morphosyntactic patterns used to generate stems from roots of a certain type. This table also specifies the syntactic properties of stems (voice and tense) generated by using the template entry. The *MainDictionary* table links the *RootsList* and *Template* tables together and specifies which entries apply to which roots.

Stems generated with these tables are unaffixed stems. The *affix_id* field links each entry to a subject pronominal affix table that uses transformation rules generating affixed stems. Although object pronominal affixes are not dealt with in this paper, they are generally agglutinating in nature and therefore cause no morphophonemic alterations to a stem. They can be added for generation or removed for analysis without affecting the stem at all.

Affixation and transformation rules are both specified using PERL regular expressions (Friedl, 2002). Regular expressions (Regexp) is an algebraic language that is used for building finite state transducers (FSTs) that accept regular languages. In the next section, Regexp is used to perform morphophonemic transformations and to generate affixed forms of stems. If generated stems are to be useful for root extraction and morphological analysis, it is essential at every stage of generation to be able to track exactly which letters are members of the root radical set, which belong to the template, and what transformations occur on the untransformed stem producing the final surface form.

3 Definition of Stem Generation

In order to be useful in analysis applications, Arabic stems need to be in a surface form which will only undergo agglutinating changes for any further mor-

phological modification. Stems should be defined in terms of the root radicals, morphosemantic and morphosyntactic template letters, and morphophonemic alterations. By doing so, inversing stem transformations becomes trivial. We require the automatic stem generator to always be aware of the origin of each of the letters in stems it generates and to be able to distinguish between letters in the original radical set or in the template string. The stem generator may then be used to compile a complete list of all affixed stems from database roots while retaining all transformation information. The resulting list of stems may then be turned into a searchable index that holds the complete morphological analysis and classification for each entry.

Since originally Arabic words can have a maximum of four root radicals, a root radical set R is defined in terms of the ordered letters of the root as follows:

$$R = \{r_{\mathbf{F}}, r_{\mathbf{M}}, r_{\mathbf{L}}, r_{\mathbf{Q}}\} \tag{1}$$

In the database, pattern, root, variant, and voice-tense ids identify a particular morphological pattern s. Templates are used to generate a stem from a root. The text of s is defined in terms of the letters and diacritics of the template in sequence ($x_1 ... x_l$) and the radical position markers or place holders ($h_{\mathbf{F}}$, $h_{\mathbf{M}}$, $h_{\mathbf{L}}$, and $h_{\mathbf{Q}}$), that indicate the positions that letters of the root should be slotted into:

$$s = x_1 x_2 ... h_{\mathbf{F}} ... h_{\mathbf{M}} ... h_{\mathbf{L}} ... h_{\mathbf{Q}} ... x_n \tag{2}$$

Stem Generator (SG) uses regular expressions as the language for compiling FSTs for morphophonemic transformations. Transformation rules take into account the context of root radicals in terms of their positions in the template and the nature of the template letters that surround them. Transformations are performed using combinations of regular expression rules applied in sequence, in a manner similar to how humans are subconsciously trained to process the individual transformations. The resulting template between one morphophonemic transformation and the next is an intermediate template. However, in order to aid the next transformation, the transformed radicals are marked by inserting their place holders before them. For example, $h_{\mathbf{F}}$ ر $h_{\mathbf{M}}$ س $h_{\mathbf{L}}$ م (**F**a**M**s**L**m**a**) is an intermediate template formed by the root radical set R ={ر, س, م} ({r, s, m}) and the morphological pattern $s = h_{\mathbf{F}}\acute{} \; h_{\mathbf{M}}\acute{} \; h_{\mathbf{L}}\acute{}$ (**F**a**M**a**L**a).

To create the initial intermediate template i_0 from the radical set R and morphological pattern s, a function $\mathrm{Regexp}(String, SrchPat, ReplStr)$ is defined to compile FSTs from regular expressions. The function accepts in its first argument a string that is tested for a match with the search pattern ($SrchPat$) in its second argument. If $SrchPat$ is found, the matching characters in $String$ are replaced with the replace string ($ReplStr$). This function is assumed to accept the standard PERL regular expression syntax.

A function, $\mathrm{CompileIntermediate}(R, s)$, accepts the radical set R and morphological pattern s to compile the first intermediate template i_0. A regular expression is built to make this transformation. It searches the morphological pattern text for radical place holders and inserts their respective radical values after them. Since Regexp performs substitutions instead of insertions, replacing each marker with itself followed by its radical value is effectively equivalent to inserting its radical value after it. Let p be a search pattern that matches all occurrences of place holders $h_{\mathbf{F}}$, $h_{\mathbf{M}}$, $h_{\mathbf{L}}$, or $h_{\mathbf{Q}}$ in the morphological pattern, then an initial intermediate form i_0 may be compiled in the following manner:

$$\begin{aligned} i_0 &= \mathrm{CompileIntermediate}(R, s) \\ &= \mathrm{Regexp}(s, p, pR_p) \\ &= \{x_1 ... h_{\mathbf{F}} r_{\mathbf{F}} ... h_{\mathbf{M}} r_{\mathbf{M}} ... h_{\mathbf{L}} r_{\mathbf{L}} ... h_{\mathbf{Q}} r_{\mathbf{Q}} ... x_n\} \end{aligned} \tag{3}$$

Let $T = \{t_1 ... t_m\}$ be the transformation rules applied on each intermediate template to create subsequent intermediate templates. Transformation rules are defined as:

$$t_j = (SrchPat_j, ReplStr_j) \tag{4}$$

A second function $\mathrm{Transform}(i, t)$ is required to perform transformations. A subsequent intermediate template i_{j+1} is the recursive result of transforming the current intermediate template i_j with the next rule t_{j+1}. Each transformation is defined as:

$$\begin{aligned} i_{j+1} &= \mathrm{Transform}(i_j, t_{j+1}) \text{ for } 0 \leq j < m \\ &= \mathrm{Regexp}(i_j, SrchPat_{j+1}, ReplStr_{j+1}) \end{aligned} \tag{5}$$

At any point in the transformation process, the current transformed state of radicals (R') and template string (s') may be decomposed from the current intermediate template as follows:

$$\text{CompileIntermediate}^{-1}(i_j) = (R', s') \quad (6)$$

To turn final intermediate template i_m into a proper stem, a regular expression is built that deletes the place holders from the intermediate template. To do this with a regular expression, the place holders matched are replaced with the null string during the matching process as follows:

$$\text{Regexp}(i_m, p, null) \quad (7)$$

Basic stems are only modified for tense and voice. Additional morphosyntactic templates or affixation rules further modify proper stems for person, gender, number, and mode. Affixation rules are regular expressions like transformation rules. However, these rules modify final intermediate templates by adding prefixes, infixes, or suffixes, or modifying or deleting stem letters. They require knowledge of the radical positions and occasionally their morphophonemic origins. Adding affixes to a stem operates on the intermediate template which retains the necessary information.

Let a be the affixation rule that is being applied to a certain intermediate template:

$$a = (SrchPat, ReplStr) \quad (8)$$

Now using the function Transform that was defined earlier, affixes are added to i_m to produce the intermediate affixed template i_{m+1}:

$$\begin{aligned} i_{m+1} &= \text{Transform}(i_m, a) \\ &= \text{Regexp}(i_m, SrchPat, ReplStr) \end{aligned} \quad (9)$$

To convert for output i_{m+1} to an affixed stem, one may remove place holders using the following:

$$\text{Regexp}(i_{m+1}, p, null) \quad (10)$$

With this definition, generated stems are described by intermediate templates. Intermediate templates retain knowledge of the current state of template and radical letters without losing the ability to recall their origins. This algorithm, therefore, would avoid guesswork in the identification of root radicals. Automatic rule-based stem generation and analysis are both facilitated by this feature of intermediate templates.

4 Stem Generation Engine

A stem generation engine may be built on the basis of the definition just advanced. The three components, *Stem Transformer*, *Affixer*, and *Slotter*, applied in sequence, make up SG. *Stem Transformer* applies the appropriate transformation rules to the morphological pattern, *Affixer* adds specific affixes to the transformed template; and *Slotter* applies the radicals to the transformed affixed template to produce the final affixed stem.

SG begins with a stem ID from the *MainDictionary* table as input to *Stem Transformer* (See Figure 1). The root and entry associated with the stem ID are used to identify the radicals of the root, the morphological pattern string, a list of transformation rules, and an affix table ID.

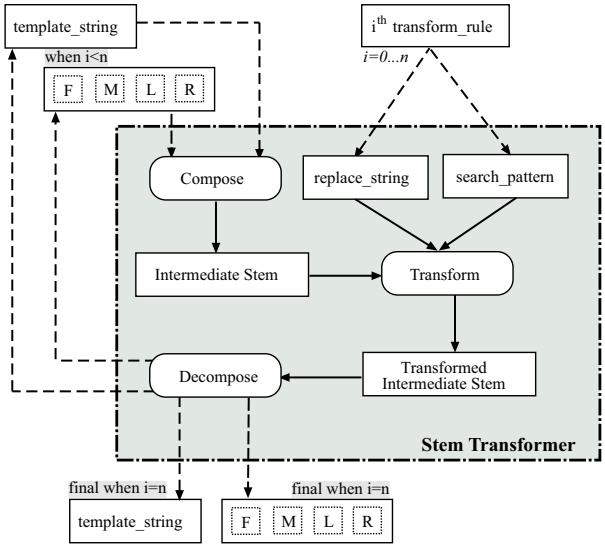


Figure 2: Stem Transformer

Stem Transformer applies transformation rules that are localised to the root radicals and letters of the template in the contexts of one another. To prepare the template and root for transformation, the engine begins by marking radicals in the template.

Stem Transformer is applied incrementally using the current radical set, the template string, and one transformation rule per pass, as in Figure 2. The output of each pass is fed back into Stem Transformer in the form of the j^{th}-rule-transformed template string and radicals, along with the $(j+1)^{th}$ transformation rule. When all rules associated with the template are exhausted, the resultant template string and radicals are output to the next phase.

To illustrate, assume the morphological pattern $s =$ ا $h_{\mathbf{F}}$ ـَ ت $h_{\mathbf{M}}$ـَ $h_{\mathbf{L}}$ـَ (Ai**F**ota**M**a**L**a), the radical set $R =$ {ذ, ك, ر} ({**@,k,r**}), and the transformation rule set $T = \{1, 12\}$.

Stem Transformer generates a proper stem using the following steps:

Equation 3 above creates the initial intermediate template when passed the radical set and morphological template, thus producing:

$$\begin{aligned} i_0 &= \text{CompileIntermediate}(R, s) \\ &= \text{ا}\, h_{\mathbf{F}}\, \text{تَ}\, \text{ذْ}\, h_{\mathbf{M}}\, \text{كَ}\, h_{\mathbf{L}}\, \text{رَ} \\ &\quad \text{(Ai}\mathbf{F}\text{@ta}\mathbf{Mka}\mathbf{Lr}\text{a)} \end{aligned}$$

The first transformation rule $t_1 = 1, t_1 \in T$ is a regular expression that searches for a ت (t) following $h_{\mathbf{F}}$ and replaces ت (t) with a copy of $r_{\mathbf{F}}$. To transform i_0 into i_1 with rule t_1, Equation 5 is used, thus producing:

$$\begin{aligned} i_1 &= \text{Transform}(i_0, t_1) \\ &= \text{ا}\, h_{\mathbf{F}}\, \text{ذَ}\, \text{ذْ}\, h_{\mathbf{M}}\, \text{كَ}\, h_{\mathbf{L}}\, \text{رَ} \\ &\quad \text{(Ai}\mathbf{F}\text{@o@a}\mathbf{Mka}\mathbf{Lr}\text{a)} \end{aligned}$$

Next, a gemination rule $t_2 = 12, t_2 \in T$ is applied to i_1. The gemination regular expression searches for an unvowelled letter followed by a vowelled duplicate and replaces it with the geminated vowelled letter. Once more, Equation 5 is used to make the transformation:

$$\begin{aligned} i_2 &= \text{Transform}(i_1, t_2) \\ &= \text{ا}\, h_{\mathbf{F}}\, \text{ذَّ}\, h_{\mathbf{M}}\, \text{كَ}\, h_{\mathbf{L}}\, \text{رَ} \\ &\quad \text{(Ai}\mathbf{F}\text{@~a}\mathbf{Mka}\mathbf{Lr}\text{a)} \end{aligned}$$

To obtain the proper stem from the intermediate template, the final intermediate template i_2 may be substituted into Equation 7:

$$\begin{aligned} Stem &= \text{Regexp}(i_2, p, null) \\ &= \text{اذَّكَرَ} \\ &\quad \text{(Ai@~akara)} \end{aligned}$$

To summarise, the final output of Stem Transformer is a root moulded into a template and a template-transformed radical set. These outputs are used as input to the affixation phase which succeeds stem transformation. Affixer, applied iteratively to the product of Stem Transformer, outputs 14 different subject-pronominally affixed

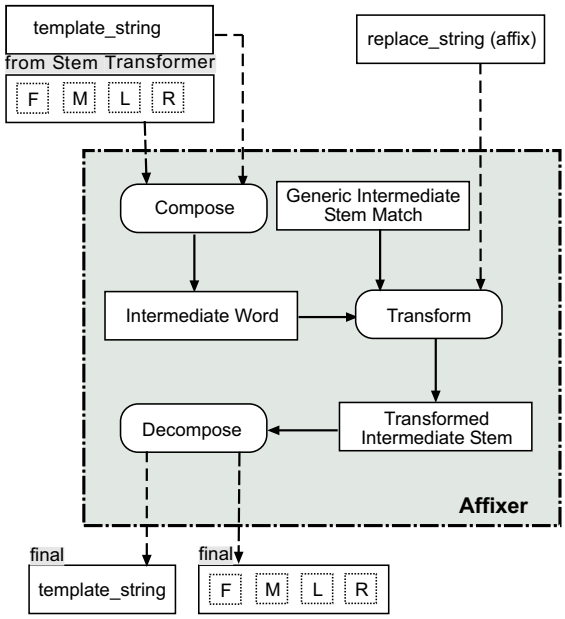


Figure 3: The Affixer Phase

morphosyntactic forms for every input except the imperative which only produces 5. There are 9 different tense-voice-mode combinations per subject pronominal affix, so most roots produce 117 affixed stems per dictionary entry. Affixer is run with different replace strings that are specific to the type of affix being produced. It modifies copies of the transformed stem from the previous phase, as in Figure 3. Using the example cited shortly before, Affixer is passed the last intermediate template i_m and the affix regular expression a. In this example, a is a regular expression that searches for $h_{\mathbf{L}}r_{\mathbf{L}}$ and replaces it with $h_{\mathbf{L}}r_{\mathbf{L}}$ تْ (**L**$r_{\mathbf{L}}$ato); this corresponds to the past active third person feminine singular affix.

Now applying Equation 9 produces:

$$\begin{aligned} i_3 &= \text{Transform}(i_2, a) \\ &= \text{ا}\, h_{\mathbf{F}}\, \text{ذَّ}\, h_{\mathbf{M}}\, \text{كَ}\, h_{\mathbf{L}}\, \text{رَ}\, \text{تْ} \\ &\quad \text{(Ai}\mathbf{F}\text{@~a}\mathbf{Mka}\mathbf{Lr}\text{ato)} \end{aligned}$$

In the last stage of stem generation, *Slotter* replaces the place holders in the transformed template with the transformed radical set, producing the final form of the affixed stem. For the example, the result of applying Equation 10 is:

$$\begin{aligned} \text{Regexp}(i_3, p, null) &= \text{اذَّكَرَتْ} \\ &\quad \text{(Ai@~akarato)} \end{aligned}$$

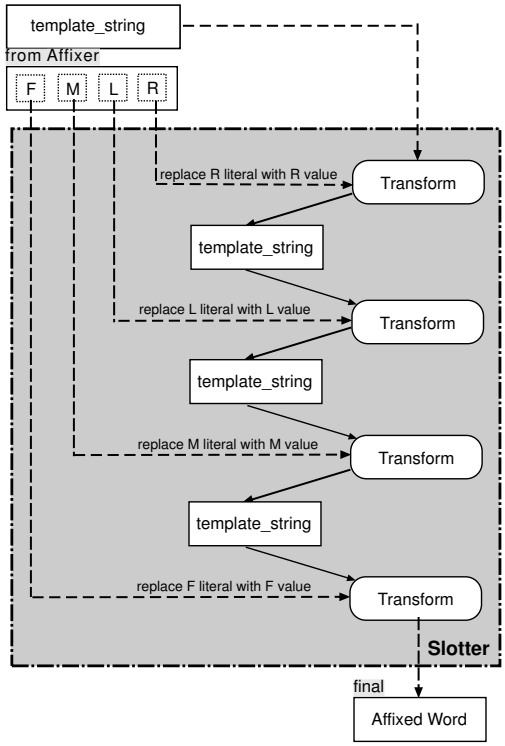


Figure 4: The Slotter Phase

5 Optimisation

Data produced for the use of SG was designed initially with no knowledge of the actual patterns and repetitions that occur with morphophonemic and affix transformation rules. In fact, SG is made to create stems this way: A root is added to a morphosemantic template, then morphosyntactic templates are applied to it, inducing in some patterns morphophonemic transformation. However, while this may be useful in many language teaching tools, it is extremely inefficient. The original data was used to discover patterns that would allow stems to be created in an optimal manner.

Following the classification in Yaghi (2004), there are 70 verb root types associated with 44 theoretically possible morphological patterns. There is an element of repetition present in the classification. In addition, the *Template* table lists sequences of rules that operate on morphological patterns in a manner similar to how native speakers alter patterns phonemically. These rules could be composed into a single FST that yields the surface form.

For example, in the previous section, the morphophonemic transformation rule set $T = \{1, 12\}$ could have been written into one rule. In its non-optimised form the rule duplicates $r_{\mathbf{F}}$ in place of ت (t) creating intermediate form ا $h_{\mathbf{F}}$ ذَ ذْ $h_{\mathbf{M}}$ كَ $h_{\mathbf{L}}$ رَ (Ai**F**@o@a**Mk**a**Lr**a) and then deletes the first of the duplicate letters and replaces it with a gemination diacritic that is placed on the second repeat letter. The resulting surface form is اذَّكَرَ (Ai@~akara). Instead, one rule could achieve the surface form by replacing the letter ت (t) in the template with a geminated ذ (@) yielding the same result.

Compiling separate regular expressions for each transformation rule is costly in terms of processing time especially when used with back-references, as SG does. Back-references group a sub-pattern and refer to it either in the search pattern or substitute string. Such patterns are not constant and are required to be recompiled for every string they are used with. It is desirable, therefore, to minimise the number of times patterns are compiled. To optimise further, the transformation may be made on the morphological pattern itself, thus producing a sound surface form template. This procedure would eliminate the need to perform morphophonemic transformations on stems.

Each template entry in the *Template* table (see Figure 1) is given a new field containing the surface form template. This is a copy of the morphological pattern with morphophonemic transformations applied. A coding scheme is adopted that continues to retain letter origins and radical positions in the template so that this will not affect affixation. Any transformations that affect the morphological pattern alone are applied without further consideration. The coding scheme uses the Roman characters **F**, **M**, **L**, and **Q** to represent place holders in the templates. Each place holder is followed by a single digit indicating the type of transformation that occurs to the radical slotted in that position. The codes have the following meanings: 0=no alteration, 1=deletion, 2=substitution, 3=gemination. If the code used is 2, then the very next letter is used to replace the radical to which the code belongs.

Take for example, the *Template* table entry for the root type 17 (all roots with F=و (w) and L=ي (y)), its morphological pattern ا$h_{\mathbf{F}}$ْتَ$h_{\mathbf{M}}$َ$h_{\mathbf{L}}$َ (Ai**F**ota**Ma**a**L**a), and its variant (ID 0́). The morphophonemic transformation rules applied to the template are T={20,12,31,34,112}. These rules correspond to the following:

- 20=change $r_{\mathbf{F}}$ to a duplicate of the next letter ت (t)
- 12=geminate duplicate letters
- 31=delete diacritic after the ي (y) in position $h_{\mathbf{L}}$
- 34=convert ي (y) to ا (A)
- 112=convert ا to ى (Y)

Surface Form	ا	$r_{\mathbf{F}}2$ت	$r_{\mathbf{M}}0$	$r_{\mathbf{L}}2$ي	(Ai	**F**2t~a	**M**0a	**L**2Y)
Affix				امَتُيْ$r_{\mathbf{L}}$	(			**L**2yotumaA)
Combined Result	ا	$r_{\mathbf{F}}2$ت	$r_{\mathbf{M}}0$	امَتُيْ$r_{\mathbf{L}}2$	(Ai	**F**2t~a	**M**0a	**L**2yotumaA)

Table 1: Surface form template aligned with an affix entry rule.

The surface form template can be rewritten as ا$h_{\mathbf{F}}2$ت$h_{\mathbf{M}}h_{\mathbf{L}}2$ى (Ai**F**2t~a**M**0a**L**2Y). This can be used to form stems such as اتَّدَى (Ait~adaY) by slotting the root {و, د, ي} ({**w**,**d**,**y**}).

The affix tables use a similar notation for coding their rules. Every affix rule indicates a change to be made to the surface form template and begins with a place holder followed by a code 0 or 2 unless the rule redefines the entire template in which case the entry begins with a 0. Radical place holders in affix rules define changes to the surface form template. These changes affect the template from the given radical position to the very next radical position or the end of the template, whichever is first.

Affix rules with code 0 following radical place holders signify that no change should be made to that section of the surface form template. However, a code 2 after a place holder modifies the surface form template in that position by replacing the letter that follows the code with the rest of that segment of the rule. Affix rules using code 2 after place holders override any other code for that position in the surface form template because affixation modifies morphophonemically transformed stems.

Creating affixed stems from templates and affixes formatted in this way becomes far more optimal. If a surface form template was specified as ا$r_{\mathbf{F}}2$ت$r_{\mathbf{M}}0r_{\mathbf{L}}2$ى (Ai**F**2t~a**M**0a**L**2Y) and it was to be combined with the affix rule امَتُيْ$r_{\mathbf{L}}2$ (**L**2yotumaA) then SG simply needs to align the affix rule with the surface form template using the place holder symbol in the affix rule and replace appropriately as in Table 1.

With the resulting affixed surface form template SG may retain the radicals of the original root where they are unchanged, delete radicals marked with code 1 and 3, and substitute letters following code 2 in place of their position holders. If the example above is used with the root {و, د, ي} ({w, d, y}), the final stem is: اتَّد يْتُمَا (Ait~a**da**yotumaA, meaning "the two of you have accepted compensation for damage").

To use the original regular expression transformations would take an average of 18000 seconds to produce a total of 2.2 million valid stems in the database. With the optimised coding scheme, the time taken is reduced to a mere 720 seconds; that is 4% of the original time taken.

6 Generated Stem Database Compiler

Figure 5: Output from the Stem Generation CGI

Once the dictionary database has been completed and debugged, an implementation of SG generates for every root, template, and affix the entire list of stems derived from a single root and all the possible template and affix combinations that may apply to that root entry. The average number of dictionary entries that a root can generate is approximately 2.5. Considering that each entry generates 117 different affixed stems, this yields an average of approximately 300 affixed stems per root. However, some roots (e.g., {ب,ت,ك} ({k,t,b})) produce 13 different entries, which makes approximately 1,500 affixed stems for each of such roots.

The generated list is later loaded into a B-Tree structured database file that allows fast stem search and entry retrieval.

A web CGI was built that uses the Stem Generation Engine to produce all affixed stems of any given root. A section of the results of this appears in Figure 5.

7 Conclusions

In this paper, we have discussed our attempt at imitating the process used by Arabic speakers in generating stems from roots. We formulated a definition of the process, facilitating an encoding of Arabic stems. The encoding represents stems in terms of their components while still allowing a simple mapping to their final surface forms. A stem's components are a root, morphosemantic and morphosyntactic templates, and any morphophonemic alterations that the stem may have underwent. In doing so, the problem has been reduced to the much smaller task of obtaining stems for the words subject to analysis, and then matching these against the surface forms of the pre-analysed stems. The encoding retains most of the information essential to stem generation and analysis, allowing us to trace the various transformations that root radicals undergo when inflected. Root extractors and morphological analysers can match an input word with a defined verb stem, then use the information in the definition to determine with certainty the stem's root and morphological pattern's meaning. The authors intend to use a similar strategy to define stems for Arabic nouns.

Mapping from words to defined stems is now much easier. The stem generation algorithm here attempts to produce a comprehensive list of all inflected stems. Any verb may be found in this list if some simple conjoin removal rules are first applied. Conjoins are defined here as single letter conjunctions, future or question particles, emphasis affixes, or object pronominal suffixes that agglutinate to a verb stem. Because conjoins may attach to a verb stem in sequence and without causing any morphological alteration, extracting stems from Arabic words becomes similar to extracting stems from English words. In fact, many of the Arabic word analysis approaches reviewed in the introduction to this paper would yield more accurate results if applied to stem extraction instead of root extraction. It would become possible to use for this purpose conventional linguistic, pattern matching, or algebraic algorithms.

The dictionary database described here can be used to form the core of a morphological analyser that derives the root of an input word, identifies its stem, and classifies its morphosemantic and morphosyntactic templates. An analyser based on these principles may be used in many useful applications, some of which are detailed in Yaghi (2004). Example applications include root, lemma based, and exact word analysis, searching, incremental searching, and concordancing.

References

S. S. Al-Fedaghi and F. S. Al-Anzi. 1989. 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.

Riyad Al-Shalabi and Martha Evens. 1998. A Computational Morphology System for Arabic. In *Proceedings of the COLING/ACL98*, pages 66–72, Montréal, Québec, Canada, August.

Kenneth R Beesley. 2001. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *ARABIC Language Processing: Status and Prospects*, Toulouse, France, July. Arabic NLP Workshop at ACL/EACL 2001.

Mohammed A. El-Affendi. 1991. An Algebraic Algorithm for Arabic Morphological Analysis. *The Arabian Journal for Science and Engineering*, 16(4B).

Mohammed A. El-Affendi. 1999. Performing Arabic Morphological Search on the Internet: A Sliding Window Approximate Matching Algorithm and its Performance. Technical report, CCIS Report King Saud University.

Jeffery E. F. Friedl. 2002. *Mastering Regular Expressions*. O'Reilly, 2nd edition, July.

Lama Hamandi, Rached Zantout, and Ahmed Guessoum. 2002. Design and Implementation of an Arabic Morphological Analysis System. In *Proceedings of the International Conference on Research Trends in Science and Technology 2002*, pages 325–331, Beirut, Lebanon.

Ibrahim Mustapha, Ahmed H. Al-Zayat, Hamid AbdelQadir, and Mohammed Ali Al-Najjar, editors. 1972. *Al-Moajam Al-Waseet*. Cairo Arab Language Academy, Cairo, Egypt.

B. Thalouth and A. Al-Dannan. 1990. A Comprehensive Arabic Morphological Analyzer Generator. In Pierre Mackay, editor, *Computers and the Arabic Language*. Hemisphere Publishing, New York.

Jim Yaghi. 2004. Computational Arabic Verb Morphology: Analysis and Generation. Master's thesis, University of Auckland.

Sane M. Yagi and Saad Harous. 2003. Arabic Morphology: An Algorithm and Statistics. In *Proceedings of the 2003 International Conference on Artificial Intelligence (IC-AI 2003)*, Las Vegas, Nevada.

Issues in Arabic Orthography and Morphology Analysis

Tim BUCKWALTER
Linguistic Data Consortium
University of Pennsylvania
Philadelphia, PA 19104 USA
timbuck2@ldc.upenn.edu

Abstract

This paper discusses several issues in Arabic orthography that were encountered in the process of performing morphology analysis and POS tagging of 542,543 Arabic words in three newswire corpora at the LDC during 2002-2004, by means of the Buckwalter Arabic Morphological Analyzer. The most important issues involved variation in the orthography of Modern Standard Arabic that called for specific changes to the Analyzer algorithm, and also a more rigorous definition of typographic errors. Some orthographic anomalies had a direct impact on word tokenization, which in turn affected the morphology analysis and assignment of POS tags.

1 Introduction

In 2002 the LDC began using output from the Buckwalter Arabic Morphological Analyzer (Buckwalter, 2002), in order to perform morphological annotation and POS tagging of Arabic newswire text. From 2002 to 2004 three corpora were analyzed and over half a million Arabic word tokens were annotated and tagged (see Table 1).[1]

Corpus	Arabic Word Tokens
AFP	123,810
Ummah	125,698
Annahar	293,035
Total	542,543

Table 1: Arabic newswire corpora

[1] The tagged AFP, Ummah, and Annahar corpora were published as "Arabic Treebank: Part 1 v 2.0" (Maamouri 2003), "Arabic Treebank: Part 2 v 2.0" (Maamouri 2004), and "Arabic Treebank: Part 3 v 1.0" (Maamouri 2004), respectively, and are available from the LDC website <http://www.ldc.upenn.edu >

The author was responsible for developing and maintaining the Analyzer, which primarily involved filling in the gaps in the lexicon and modifying the POS tag set in order to meet the requirements of treebanking efforts that were performed subsequently at the LDC with the same annotated and POS-tagged newswire data.

2 Lessons from the AFP corpus

During the tagging of the AFP data, the first corpus in the series, the Buckwalter Analyzer was equipped to handle basic orthographic variation that often goes unnoticed because it is a common feature of written Arabic (Buckwalter, 1992). This orthographic variation involves the writing (or omission) of *hamza* above or below *alif* in stem-initial position, and to a lesser extent, the writing (or omission) of *madda* on *alif*, also in stem-initial position. In both cases use of the bare *alif* without *hamza* or *madda* is quite common and goes by unnoticed by most readers. What took the LDC morphology annotation team by surprise was to find that in the AFP data the common omission of *hamza* in this environment had been extended to stem-medial and stem-final positions as well, as seen in the following words from that corpus: بدات تايید متاخر تاسیس تستانف رات لتاكيد بشان.

This type of orthographic variation was not attested to the same extent in the two subsequent corpora, Ummah and Annahar, which leads us to conclude that some orthographic practices might be restricted to specific news agencies. It is important to note that most of the native Arabic speakers who annotated the AFP data using the output from the Analyzer did not regard these omissions of *hamza* on *alif* in stem-medial and stem-final positions as orthographic errors, and fully expected the Analyzer to provide a solution.

3 Lessons from the Ummah corpus

During the tagging of the Ummah data, a different set of orthographic issues arose. Although the Buckwalter Analyzer was equipped to handle so-called "Egyptian" spelling (where word-final *ya'* is spelled without the two dots, making it

identical to *alif maqsura*), the Ummah corpus presented the LDC annotation team with just the opposite phenomenon: dozens of word-final *alif maqsura*'s spelled with two dots.[2] Whereas some of the affected words were automatically rejected as typographical errors (e.g., القري الأعلي موسي متي أخري), others where gladly analyzed at face value (e.g., لدي إلي علي). Unfortunately, this led to numerous false positive analyses: for example علي was analyzed as '*ali* and '*alayya*, but not as '*ala*. Initially, these words were tagged as typographical errors, but their pervasiveness led the LDC team to reconsider this position, upon which the author was asked to modify the Analyzer algorithm in order to accommodate this typographic anomaly. As a result, all words ending in *ya*' were now re-interpreted as ending in either *ya*' or *alif maqsura*, and both forms were analyzed, as seen in the following (abridged) output:[3]

```
<token_Arabic>علي
  <variant>Ely
    <solution>
      <lemmaID>EalaY_1</lemmaID>
      <pos>Ealay/PREP+ya/PRON_1S</pos>
      <gloss>on/above + me</gloss>
    </solution>
    <solution>
      <lemmaID>Ealiy~_1</lemmaID>
      <voc>Ealiy~N</voc>
      <pos>Ealiy~/ADJ+N/CASE_INDEF_NOM</pos>
      <gloss>supreme/high + [indef.nom.]</gloss>
    </solution>
    <solution>
      <lemmaID>Ealiy~_2</lemmaID>
      <voc>Ealiy~N</voc>
      <pos>Ealiy~/NOUN_PROP+N/CASE_INDEF_NOM</pos>
      <gloss>Ali + [indef.nom.]</gloss>
    </solution>
  </variant>
  <variant>ElY
    <solution>
      <lemmaID>EalaY_1</lemmaID>
      <voc>EalaY</voc>
      <pos>EalaY/PREP</pos>
      <gloss>on/above</gloss>
    </solution>
  </variant>
</token_Arabic>
```

4 Lessons from all three corpora

The Annahar corpus presented no orthographic surprises, or at least nothing that the LDC annotation team had not seen before. The Annahar data did contain some additional orthographic features that we now identify as being common to all three corpora, as well as corpora outside the set we have annotated at the LDC.

The first orthographic feature relates to the somewhat free interchange of stem-initial *hamza* above *alif* and *hamza* below *alif*. With some lexical items the orthographic variation simply reflects variation in pronunciation: for example, both '*isbaniya* (with *hamza* under *alif*) and '*asbaniya* (with *hamza* above *alif*) are well attested. But in cases involving other orthographic pairs, more interpretations are possible. Take, for instance, what we called the "*qala 'anna*" problem. This problem was identified after numerous encounters with constructions in which *qala* was followed by '*anna* rather than '*inna*, and for no apparent linguistic reason. Initially this was treated as a typographical error, but again, its pervasiveness forced us to take a different approach.

One solution we considered was to modify the Analyzer algorithm so that instances of stem-initial *hamza* on *alif* would also be treated as possible instances of *hamza* under *alif*, very much in the spirit of the approach we used for dealing with the *alif maqsura* / *ya*' free variation cited earlier. However, there is compelling evidence that the orthography of *hamza* in stem-initial position is a fairly reliable indication of the perceived value of subsequent short vowel: *a* or *u* for *hamza* above *alif*, and *i* for *hamza* below *alif*. In other words, there is no free variation. The decision was taken to regard "*qala 'anna*" constructions as grammatically acceptable in MSA.[4]

5 Concatenation in Arabic orthography

The second, and more serious, orthographic anomaly we encountered in all three corpora is what we call the problem of Arabic "run-on" words, or free concatenation of words when the word immediately preceding ends with a non-connector letter, such as *alif*, *dal*, *dhal*, *ra*, *za*, *waw*, *ta marbuta*, etc.

The most frequent "run-on" words in Arabic are combinations of the high-frequency function words *la* and *ma* (which end in *alif*) with following perfect or imperfect verbs, such as *la-yazal*, *ma-yuram*, and *ma-zala* (مازال مايرام لايزال). The *la* of "absolute negation" concatenates freely with nouns, as in *la-budda*, *la-shakka* (لاشك لابد). It can be argued that these are lexicalized collocations, but their spelling with intervening space (لا يزال –

[2] It is not entirely clear whether these "dotted" *alif maqsura*'s were produced by human typists or by an encoding conversion process gone awry. It is possible that the original keyboarding was done on a platform where word-final *ya*' and *alif maqsura* are displayed via visually identical "un-dotted" glyphs, so it makes no difference which of the two keys the typist presses on the keyboard: both produce the same visual display, but are stored electronically as two different characters.

[3] A key to the transliteration scheme used by the Analyzer can be found at <http://www.ldc.upenn.edu/myl/morph/buckwalter.html>

[4] Badawi, Carter and Gully regard "*qala 'anna*" constructions as grammatical but restricted to contexts "where the exact words of the speaker are not used or reported" (Badawi, Carter and Gully 2004, p. 713). This assertion could be investigated in the LDC corpora.

لا بد – ما زال) is just as frequent as their spelling in concatenated form.

Proper name phrases, especially those involving the word *'abd* (عبدالرحمن عبدالله) are also written either separately or in concatenated form. Part of the data annotation process at the LDC involves assigning case endings to tokenized words, but there is currently no mechanism in the Analyzer to assign two case endings (or several pairs of POS tags) to what is being processed as a single word token. As a result of this, the phrase '*abd allah* is assigned a single POS tag and case ending when it is written in concatenated form, but two POS tags and two case endings when written with intervening space.

The problem of assigning more than one case ending and POS tag to concatenations is more obvious in fully lexicalized concatenations such as *khamsumi'atin*, *sittumi'atin*, *sab'umi'atin*, etc (سبعمائة – ستمائة – خمسمائة). When these numbers are written with intervening space (ست مائة – خمس مائة – سبع مائة), two case endings and two POS tags are assigned by the Analyzer. But when they are written in concatenated form only one case ending and POS tag is assigned, and the "infixed" case ending of the first token is left undefined: *khamsmi'at*^{in}, *sittmi'at*^{in}, *sab'mi'at*^{in}, etc.[5]

So far we have discussed relatively controlled concatenation, involving mostly high-frequency function words and lexicalized phrases. But concatenation extends beyond that to random combinations of words—the only requirement being that the word immediately preceding end with a non-connector letter. These concatenations are fairly frequent, as attested by their Google scores (see Table 2).

It is important to note that these concatenations are not immediately obvious to readers due to the characteristics of proportionally spaced Arabic fonts. Most of the native readers of Arabic at the LDC did not consider concatenations such as these to be typographical errors. Their logic was best expressed in the statement: "I can read the text just fine. Why can't the Morphological Analyzer?"

Concatenation	Google Frequency
مديرعام	846
وزيرالخارجية	719
ملياردولار	162
الدكتورمحمد	158
عضومجلس	138
وقدتم	130
واشارالى	99
كماتم	77
عددكبير	54

Table 2: Arabic Concatenations and their Google Frequencies (sample taken March 25,2004)

6 Conclusion

There are several levels of orthographic variation in Arabic, and each level calls for a specific response to resolve the orthographic anomaly. It is important that the output analysis record which method was used to resolve the anomaly. The methods used for resolving orthographic anomaly range from exact matching of the surface orthography to various strategies of orthography manipulation. Each manipulation strategy carries with it certain assumptions about the text, and these assumptions should be part of the output analysis. For example, an analysis of علي obtained by exact matching in a text known to contain suspicious word-final *ya*'s (that may be *alif maqsura*'s) does not have the same value as an analysis of the same word, using the same exact matching, but in a text where word-final *ya*'s and *alif maqsura*'s display normal character distribution frequencies.

The problem of run-on words in Arabic calls for a reassessment of current tokenization strategies, including the definition of "word token" itself.[6] It should be assumed that each input string represents one or more potential word tokens, each of which needs to be submitted individually for morphology analysis. For example, the input string فقدتم can be segmented as a single word token, yielding two morphological analyses (*faqad-tum* and *fa-qud-tum*) or it can be segmented as two word tokens (*fqd tm*), yielding several possible analysis pairs (*faqada / fuqida / faqd / fa-qad + tamma*).

[5] We regard these as "fully lexicalized" concatenations because the first of the two constituent tokens ends in a connector letter. In other word, their concatenation is deliberate and not an accident of orthography.

[6] By "tokenization" we mean the identification of orthographically valid character string units that can be submitted to the Analyzer for analysis. The Analyzer itself performs a different kind of "tokenization" by identifying prefixes and suffixes that are bound morphemes but which may be treated as "word tokens" in syntactic analysis.

Syntactic analysis would be needed for determining which morphology analysis is most likely the correct one for each tokenization (*fqdtm* and *fqd tm*).

7 Acknowledgements

Our thanks go to the Arabic annotation team at the LDC, especially the team of native speaker informants that provided the author with daily feedback on the performance of the Morphological Analyzer, especially in areas which led to a reassessment and better understanding of orthographic variation, as well as tokenization and functional definitions of typographical errors.

References

Elsaid Badawi, M.G. Carter, and Adrian Wallace. 2004. *Modern Written Arabic: A Comprehensive Grammar.* Routledge, London.

Tim Buckwalter. 1992. "Orthographic Variation in Arabic and its Relevance to Automatic Spell-Checking," in *Proceedings of the Third International Conference on Multilingual Computing (Arabic and Roman Script),* University of Durham, U.K., December 10-12, 1992.

Tim Buckwalter. 2002. *Buckwalter Arabic Morphological Analyzer Version 1.0.* Linguistic Data Consortium, catalog number LDC2002L49 and ISBN 1-58563-257-0. < http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49 >

Mohamed Maamouri, et al. 2003. *Arabic Treebank: Part 1 v 2.0.* Linguistic Data Consortium, catalog number LDC2003T06 and ISBN: 1-58563-261-9. < http://www.ldc.upenn.-edu/Catalog/CatalogEntry.jsp?catalogId=LDC2003T06 >

Mohamed Maamouri, et al. 2004. *Arabic Treebank: Part 2 v 2.0.* Linguistic Data Consortium, catalog number LDC2004T02 and ISBN: 1-58563-282-1. < http://www.ldc.upenn.-edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004T02 >

Mohamed Maamouri, et al. 2004. *Arabic Treebank: Part 3 v 1.0.* Linguistic Data Consortium, catalog number LDC2004T11 and ISBN: 1-58563-298-8. < http://www.ldc.upenn.-edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004T11 >

Finite-State Morphological Analysis of Persian

Karine Megerdoomian

Inxight Software, Inc.
500 Macara Avenue
Sunnyvale, CA 94085, USA
karinem@inxight.com

University of California, San Diego
Linguistics Department
9500 Gilman Drive, #0108
La Jolla, CA 92093, USA
karinem@ling.ucsd.edu

Abstract

This paper describes a two-level morphological analyzer for Persian using a system based on the Xerox finite state tools. Persian language presents certain challenges to computational analysis: There is a complex verbal conjugation paradigm which includes long-distance morphological dependencies; phonological alternations apply at morpheme boundaries; word and noun phrase boundaries are difficult to define since morphemes may be detached from their stems and distinct words can appear without an intervening space. In this work, we develop these problems and provide solutions in a finite-state morphology system.

1 Introduction

This paper describes the design of a two-level morphological analyzer for Persian developed at Inxight Software, based on Xerox finite-state technology (Beesley and Karttunen, 2001), by focusing on some of the issues that arise in a computational analysis of the language.

Persian morphology raises some interesting issues for a computational analysis. One of the main challenges of Persian resides in the tokenization of the input text, since word boundaries are not always respected in written text. Hence, morphemes may appear detached from their stems while distinct tokens may be written without an intervening space. Furthermore, the use of the Arabic script and the fact that short vowels are not written and capitalization is not used create ambiguities that impede computational analysis of text. Persian includes *complex tokens* whereby two distinct part of speech items may be joined; these attaching elements (e.g., prepositions, pronominal clitics or verbs) should be treated as inflectional morphemes in the morphological analyzer. Persian does not have the problems that have been observed in Semitic languages such as the template-based morphology of Arabic, and is in general more concatenative. However, the verbal conjugation consists of a complex paradigm, which includes long-distance dependencies that may be problematic for a linear approach depending solely on surface forms. Finally, the phonetic representation of Persian nominals directly affects the phonological alternations applying at morpheme boundaries; however, the orthographic realization of certain words may not reflect their phonetics and require special manipulations to eliminate the ambiguities.

Although there have been some significant studies in the area of parsing and syntactic analysis for Persian, very little work has been done on computational morphology in this language. In this paper, we elaborate on some of the challenges presented by a morphological analysis of Persian and discuss the solutions provided with a two-level finite-state formalism.

2 System Description

The Persian system is developed using Xerox Finite-State Technology. The lexicons and morphological rules are written in the format of *lexc*, which is the lexicon compiler (Karttunen and Beesley, 1992). The lexicon and grammar are compiled into a finite-state transducer (fst) where the lower side consists of the input string and the upper side provides the baseform of the word with associated morphosyntactic features. In this system, the fsts for each part of speech category are created separately and then composed. Similarly, phonological rules are composed on the relevant fst, thus performing the required phonetic and phonological alternations on the word forms. The composition of all the part of speech transducers with the rules results in the final lexical transducer used for morphological analysis. Since all intermediate levels disappear during a composition, the final transducer consists of a single two-level fst with surface strings in the bottom and the morphological output on the top.

Consider the simple lexc example below. This lexc consists of three small LEXICONs, beginning with the one named Root, which marks the start of the network. The lexicon class named Root

includes three entries and each entry consists of a *form* and a *continuation class*.

```
LEXICON Root
dog  Noun ;
cat  Noun ;
laugh     Verb ;

LEXICON Noun
+Plural:s   # ;
+Singular:0 # ;

LEXICON Verb
+Present:s  # ;
+Past:ed    # ;
+Gerund:ing # ;
            # ;  !empty string
```

The forms, such as 'dog', are interpreted by the lexc as a regular expression as in {d o g}. Continuation classes are used to account for word-formation by capturing morphotactic rules. In the example under consideration, the string 'dog' is followed by the continuation class Noun. As the Noun lexicon shows, the rule allows 'dog' to be followed either by the morpheme 's' or by a null morpheme represented as '0'. The Noun continuation class maps the lower string 's' to the +Plural tag on the upper side of the two-level transducer. Similarly, the Verb continuation class allows the concatenation of the verbal stem 'laugh' with the various inflectional morphemes.

The Persian morphological analyzer at Inxight currently consists of about 55,000 stem forms, including multiword tokens, and a system of rules that identify the baseform of each token. Examples of the output of the morphological analyzer are shown below where the left hand side represents the lower input string and the right hand side is the upper side output[1]:

مسافرین 'travelers'
msâfryn → *msâfr*+Noun+Pl
رفت 'he/she left'
rft → *rftn*+Verb+Ind+Pret+3P+Sg
وکیلست 'he/she is a lawyer'
vkylst→*vkyl*+Noun>*bvdn*+Verb+Ind+Pres+3P+Sg

The rules are written as regular expressions and are represented as continuation paths within the lexc grammar. The morphological analyzer covers all main features of the Persian language with full verbal conjugation and nonverbal inflection, including irregular morphology. In addition, about twenty phonological rules are used to capture the various surface word forms and alternations that occur in the language. Common Proper Nouns are also recognized and tagged.

3 Challenges of the Persian System

This section outlines some of the main issues that arise in a computational analysis of Persian text and presents the approach adopted in the current finite-state system. Comparisons are made with past work on Persian morphological analyzers when relevant.

Persian is an affixal system consisting mainly of suffixes and a number of prefixes appearing in strict morphotactic order. The nonverbal paradigm consists of a relatively small number of affixes marking number, indefiniteness or comparatives, but the language has a complete verbal inflectional system, which can be obtained by the various combinations of prefixes, stems, person and number inflections and auxiliaries.

3.1 Nonverbal Morphology

The Arabic script used in Persian distinguishes between the attached and unattached (or final) forms of the characters. Thus, letters in a word are often connected to each other, whereas all but six characters have a final form if they appear at the end of a word or token. Thus, most characters have a different form depending on their position within the word and the final forms can therefore be used to mark word boundaries. But as we will see in this section, these boundaries are not without ambiguity.

Detached inflectional morphemes. The Persian writing system allows certain morphemes to appear either as bound to the host or as free affixes – free affixes could be separated by a final form character or with an intervening space. The three possible cases are illustrated for the plural suffix *hâ* (ها) in *flsTyny hâ* (فلسطینی ها) 'Palestinians' and the imperfective prefix *my* (می) in *my rvnd* (می روند) 'they are going'. In these examples, the tilde (~) is used to indicate the final form marker which is represented as the control character \u200C in Unicode (also known as the zero-width non-joiner). As shown, the affixes may be attached to the stem, they may be separated with the final form control marker, or they can be detached and appear with the intervening control marker as well as a whitespace. All of these surface forms are attested in various Persian corpora.

[1] Unless otherwise specified, the Persian examples are direct transliterations of the Persian script and do not include short vowels, since that would require disambiguation of word senses and is beyond the scope of the current application. For issues in automatic diacritization of Arabic script-based tests see (Vergyri and Kirchhoff, 2004) in this volume.

Attached	Final Form	Intervening Space
flsTynyhâ	*flsTyny~hâ*	*flsTyny~ hâ*
myrvnd	*my~rvnd*	*my~ rvnd*

In his two-level morphological analyzer, (Riazati, 1997) is unable to analyze the detached affixes and decides to treat these elements in syntax. Thus, the two surface realizations of morphemes such as the plural *hâ* are analyzed in different levels of the system (the attached version in the morphological analyzer and the detached form in the syntactic parser). In the unification-based system developed at CRL (Megerdoomian, 2000), a post-tokenization component is used to join the detached morpheme to the stem, separated by the control character. The morphological grammar is then designed to recognize both surface forms.

The advantage of the finite-state system described here is the ability to process multiword tokens in the analyzer. Thus, by treating the final form character (the zero-width non-joiner) as a space in the tokenization rules, we are able to analyze the detached morphemes in Persian as part of multiword tokens within the lexc grammar module. This allows us to treat both forms uniformly in the morphological analyzer and there is no need for a preprocessing module or for delaying the analysis of the detached morphemes to the syntactic level.

Complex tokens. "Complex tokens" refer to multi-element forms, which consist of affixes that represent a separate lexical category or part of speech than the one they attach to. As in languages such as Arabic and Hebrew, Persian also allows attached word-like morphemes such as the preposition *bh* (به) (*b-* in attached form), the determiner *ayn* (این), the postposition *râ* (را), or the relativizer *kh* (که), that form such complex tokens and need to be analyzed within the morphological analyzer. Similarly, a number of pronominal or verbal clitic elements may appear on various parts of speech categories, giving rise to complex tokens. The examples below illustrate some of these complex constructions where two distinct part of speech items appear attached. The word-like affixes are shown in bold in the examples below.

(i) ***b**eqydh* *Smâ* بعقیده شما
to+opinion you
'in your opinion'

(ii) ***ayn**kâr* اینکار
this+work
'this work'

(iii) anqlaby-tryn-ha-**ySan-nd** انقلابیترینهایشانند
revolutionary+Sup+Plur+**Pron.3pl+Cop.3pl**
'they are the most revolutionary ones'

To account for these cases in the Persian system, the different part of speech items are analyzed within the morphological analyzer and they are separated with an angle bracket as shown below for *ktabhayman* (کتابهایمان) 'our books' and *beqydh* (بعقیده) 'to+opinion'.

ktabhayman
→*ktab*+Noun+Pl>*av*+Pron+Pers+Poss+1P+Pl+Clit
beqydh
→ *bh*+Prep< *eqydh* +Noun+Sg

The angle brackets are used to distinguish these elements from regular inflectional morphemes since the distinct part of speech information may be needed at a later stage of processing, e.g., for parsing or machine translation. Each word-like prefix is presented by its stem form: *av* (او) 'he/she' for the pronominal clitic and *bh* (به) 'to' for the baseform of the preposition. This stem form is then followed by the relevant morphosyntactic tags. If the information is not required, as in the case of certain information retrieval applications, the elements separated by the angle brackets can easily be stripped off without losing the information of the content carrying category, namely the noun in these examples.

In certain cases, two distinct syntactic categories may appear without an intervening space even though they are not attached. For instance, the preposition *dr* (در) 'in' ends in the character 'r' which does not distinguish between a final form and an attached form. Sometimes *dr* appears without a space separating it from the following word and the tokenizer is not able to segment the two words since there is no final form to mark the word boundary. Similarly, in many online corpora sources, the coordination marker *v* (و) 'and' appears juxtaposed with the following word without an intervening space; and since the letter 'v' does not distinguish between a final and attached form, the tokenizer cannot determine the word boundary. These common words that often appear written without an intervening space, though not actually inflectional morphemes, are treated as prefixes in the system as illustrated below:

vgft → *v*+Coord< *gftn*+Verb+Pret+3P+Sg وگفت
drdftr → *dr*+Prep< *dftr*+Noun+Sg دردفتر

Phonetics & Phonological Rules. In Persian, the form of morphological affixes varies based on the ending character of the stem. Hence, if an

animate noun ends in a consonant, it receives the plural morpheme *–ân* as in *znân* (زنان) 'women'. If the animate noun ends in a vowel, the glide 'y' is inserted between the stem and the plural morpheme as in *gdâyân* (گدایان) 'the poor'. Similarly, for animate nouns that end in a silent 'h' (i.e., the letter 'h' which is pronounced as *é*), they take the morpheme *–gân* as in *frSth* (فرشته) → *frStgân* (فرشتگان) 'angels'.

A problem arises in Persian with characters that may be either vowels or consonants and cannot be analyzed correctly simply based on the orthography. For instance, the character 'v' is a consonant in *gâv* (گاو) 'cow' (pronounced 'gaav') but a vowel in *dânSJv* (دانشجو) 'university student' (pronounced 'daneshjoo'). The character 'h' is pronounced as a consonant in *mâh* (ماه) 'moon' but as a vowel in *bynndh* (بیننده) 'viewer' (pronounced 'binandé'). Similarly, 'y' is a glide in *r'ay* 'vote' but a vowel in *mâhy* (ماهی) 'fish' (pronounced 'maahee'). Hence, it is clear that in Persian, the orthographic realization of a character does not necessarily correspond to the phonetic pronunciation, yet phonological alternations of morphemes are sensitive to the phonetics of stems.

In the finite-state lexicon, the nonverbal and closed class lexical items are separated based on their final character, i.e., whether they end in a consonant or a vowel, and word boundary tags are used to determine the relevant phonological alternations. In particular, the words ending in a vowel sound are marked with a word boundary tag ^WB. Hence, the words *dânSJv, bynndh* and *mâhy* will be marked with a ^WB tag but not those ending in the consonant pronunciation of the same characters, namely *gâv*, *mâh* and *r'ay*. This allows us to convert the nominal endings of these words to their phonetic pronunciation rather than maintaining their orthographic realization, helping us disambiguate phonological rules for nominal affixes.

The words tagged with the boundary marker ^WB undergo phonetic alternations which convert the ending characters 'v', 'h' and 'y' to 'u', 'e' and 'i', respectively, in order to distinguish vowels and consonants when the phonological rules apply. Thus, after the phonetic alternations have applied, the word *mâh* ending in the consonant 'h' is transliterated as [mah] while the word *bynndh* ending in the vowel or silent 'h' is represented as [bynnde].

Once the ending vowel and consonant characters have been differentiated orthographically, the phonological alternation rules can apply correctly. We mark morpheme boundaries in the lexc with the tag ^NB. This permits the analysis routine to easily locate the area of application of the phonological alternations when the rules are composed with the lexicon transducer. One such phonological rule for the animate plural marker *-ân* is exemplified below:

```
define plural [e %^NB → g || _ a n];
```

This regular expression rule indicates that the word ending in the vowel 'e' and followed by a morpheme boundary marker is to be replaced by 'g', in the context of the plural morpheme 'an'. This rule captures the phonological alternation for *bynndh* (بیننده) 'viewer' → *bynndgân* (بینندگان) 'viewers'.

Thus, since the phonetic representation of Persian nouns and adjectives plays a crucial role in the type of phonological rule that should apply to morpheme boundaries, we manipulate the orthographic realization of certain words in order to eliminate the ambiguities that may arise otherwise.

Past morphological analysis systems have either not captured the pronunciation-orthography discrepancy in Persian thus not constraining the analyses allowed, or they have preclassified the form of the morpheme that can appear on each token. The advantage of the current system is that, by using phonological rules that apply across the board at all morpheme boundaries, we can capture important linguistic generalizations. For instance, there is no need to write three distinct plural rules to represent the various surface forms of the plural suffix *–ân* (namely, *-ân, -gân,* and *–yân*). Instead, we can write one single rule adding the *–ân* morpheme and apply phonological rules that can also apply to the boundaries for the pronoun clitic, indefinite, 'ezafe' and relativizing enclitic morphemes, providing a very effective linguistic generalization.

3.2 Verbal Paradigm

The inflectional system for Persian verbs is quite complex and consists of simple forms and compound forms; the latter are forms that require an auxiliary verb. There are two stems used in the formation of the verbal conjugation, which may combine with prefixes marking the imperfective, negation or subjunctive, person and number inflections, suffixes for marking participle forms, and the causative infix. Certain tenses also use auxiliaries to form the perfect forms, the future tense or the passive constructions.

Two stems. One of the intricacies of the Persian verbal system (and of Indo-Aryan verbal systems in general) is the existence of two distinct stem types used in the formation of different tenses: The *present stem* is used in the creation of

Form	Tense	Prefix	Stem	Inflection	Auxiliary
mygryzd می گریزد	Present	Imperfective *my*	Present *gryz*	Present.3sg *d*	---
mygryxt می گریخت	Imperfect	Imperfective *my*	Past *gryxt*	Past.3sg ‘ ‘	---
mygryxth ast می گریخته است	Compound Imperfect	Imperfective *my*	Past *gryxt*	Participle *h*	Present be.3sg) *and*
bgryz بگریز	Imperative	Subjunctive *b*	Present *gryz*	Imperative.2sg ‘ ‘	---

Table 1: Long-distance dependency between prefix and personal inflection

the present tense, the simple subjunctive, the imperative and the present participle. On what is known as the *past stem* are formed the preterite, the imperfect, the past participle and past compounds. Furthermore, all infinitives and future tenses are built on the past stem while all causatives, regardless of tense, are created on the present stem. For computational purposes, the two stems are treated as distinct entities because they often have different surface forms and cannot be derived from each other. Two examples are given below for *krdn* (کردن) and *gryxtn* (گریختن) in the actual pronunciation[2]:

Infinitival	Present Stem	Past Stem	
kardan	*kon*	*kard*	‘to do/make’
gorixtan	*goriz*	*gorixt*	‘to flee’

Since the infinitival or citation form of the verbs is built on the past stem, the verbal finite-state transducer has to produce the past stem on the upper side, allowing the derivation of the infinitive. A problem arises when the input string is the present stem form as in the present tense *my gryznd* (می گریزند) ‘they are fleeing’. In this instance, we would need to output the past stem form of the verb, namely *gryxt* (گریخت). In order to capture the association between the present and past stems in Persian, we link these forms in the verbal lexicon by allowing all present stems to map to the past stem form in the upper side of the transducer, as illustrated in the first continuation class below. In addition, the same verbs have to be listed in a different lexical continuation class with the past stems alone (i.e., past stem on both lower and upper sides) in order to analyze the tenses formed on the past stem of the verb such as the imperfect *my gryxtnd* (می گریختند) ‘they were fleeing’.

```
LEXICON PresentStem
gryxt:gryz  VerbReg ;  ! to flee
nvSt:nvys   VerbReg ;  ! to write
aftad:aft   VerbReg ;  ! to fall

LEXICON PastStem
gryxt   InfBoundary ;  ! to flee
nvSt    InfBoundary ;  ! to write
aftad   InfBoundary ;  ! to fall
```

In both cases the upper side past stem string is marked with a delimiter tag ^INF which is later mapped to ‘n’, forming the surface form of the infinitive. The resulting stem form for the finite verb *my gryznd* (می گریزند) ‘they are fleeing’ is thus the infinitival *gryxtn* (گریختن) ‘to flee’.

Long-distance dependencies[3]. As can be seen in the examples given above for the verb *gryxtn* (گریختن) ‘to flee’, the prefix *my-* (می) cannot be used to distinguish the tense of the verbal entry since it is used in the formation of the present, the imperfect or the compound imperfect. In order to decide whether *my* is forming e.g., the present tense or the past imperfect, the stem and final inflection need to be taken into account. Thus, if *my* is attached to the present stem, it forms the regular present tense forms but if it is attached to the past stem, then it gives rise to either the simple imperfect or the compound imperfect, depending on the final inflection forms (see Table 1). Similarly, the imperative inflection can only appear on a present stem with the subjunctive prefix ‘b’, as shown in *bgryz* (بگریز) in Table 1, whereas only the present inflection can be used if

[2] Note that in Persian, the short vowels such as *o,a,e* are not generally transcribed, hence the direct transliteration of the examples would be

krdn kn krd ‘to do, to make’
gryxtn gryz gryxt ‘to flee’

[3] See for instance (Sproat, 1992; pages 91-92) for a description of the issue raised by “morphological long-distance dependencies” in finite-state models of morphology.

the imperfective prefix 'my' is used, as shown with *my gryzd* (می گریزد) .

Accounting for the long-distance dependency between the prefix and the personal inflection in Persian in a finite-state two-level morphological analyzer leads to very complex paths and continuation class structures in the lexical grammar. Also, using filters to capture long-distance dependencies can sometimes largely increase the size of the transducer. Since there exist several cases of interdependencies between non-adjacent morphemes in Persian verb formation, we have opted to keep a simpler continuation class structure in the lexc grammars and to instead take advantage of *flag diacritics* and their unification process.

Flag diacritics are multicharacter symbols and can be used within the lexc grammar to permit the analysis routines to use the information provided in terms of feature-value settings to constrain subsequent paths. Hence, whether a transition to the following path would apply depends on the success of the operation defined by the flag diacritic. In essence, the flag diacritic allows the system to perform a unification of the features set in the analysis process. Xerox finite state technology includes a number of different flag diacritic operators but the only one used in this Persian system is the U-type or the Unification flag diacritic. The template for the format of these flags is as follows: @U.feature.value@. Flag diacritics are used to keep the fst small and yet be able to apply certain constraints, in particular when dealing with interdependencies between non-adjacent morphemes within a word.

For example, to capture the choice of the imperative vs. the present tense inflection based on the prefix that appears on the present stem of the verb, we use a flag diacritic with the attribute PFXTYP (PrefixType) which is then set to IMP (for imperfective) or SUB (for subjunctive). This flag diacritic is set when the prefixes are read and they are unified with the PFXTYP flags at the lexical class defining the personal inflectional paradigm. If the values of the PFXTYP flag diacritic match at this point, unification takes place allowing the concatenation of the prefix and present stem combination with the personal inflection.

Similarly, the agentive, infinitive and participial forms can be formed only if there is no prefix at all on the verbal stem. This is captured by the flag diacritic attribute PFX, which has the two possible values PRESENT and ABSENT. Thus, the lexc rule for the Infinitive, for instance, requires that the PFX flag's value be set to ABSENT. This, in effect, captures the fact that *mygryxtn* (*my* 'imperfective' + *gryxt* 'past stem' + *n* 'infinitive marker') is not a valid form since the infinitive marker *–n* can only appear on a past stem that lacks an overt prefix.

4 Evaluation

The lexicon used in the Inxight system currently consists of 43,154 lemmas, which include nouns, adjectives, verbs, adverbs and closed class items. In addition, there are about 12,000 comomon proper noun entities listed in the lexicon. The system also recognizes date, number and internet expressions.

The current Persian morphological analyzer has a coverage of 97.5% on a 7MB corpus collected mostly from online news sources. The accuracy of the system is about 95%. The unanalyzed tokens are often proper nouns or words missing from the lexicon. In addition, colloquial forms of speech are not covered in the current system.

The finite state transducer consists of 178,452 states and 928,982 arcs before optimization. And the speed of the analyzer is 20.84 CPU time in seconds for processing a 10MB file executed on a modern Sun SparcStation.

5 Conclusion

This paper describes some of the challenges encountered in a computational morphological analysis of Persian and discusses the solutions proposed within the finite state system developed at Inxight Software based on the Xerox Finite State Technology. The approaches adopted are compared with past systems of Persian whenever relevant. The paper presents the problems arising from detached inflectional morphemes, as well as attached word-like elements forming complex tokens, the discrepancy between orthography and phonetics in application of phonological rules, and the interdependency between non-adjacent morphemes in a word. In each case, it was argued that methods adopted from the finite-state calculus can capture linguistic generalizations and reduce the transducer to a manageable and commercially viable size.

6 Acknowledgements

We gratefully acknowledge the help and support provided by the development team at Inxight Software and the insightful suggestions of the members of the Lingware group. I would also like to thank the anonymous reviewers for their detailed comments.

References

Mohammad-Reza Bateni. 1995. *Towsif-e Sakhteman-e Dastury-e Zaban-e Farsi* [Description of the Linguistic Structure of Persian Language]. Amir Kabir Publishers, Tehran, Iran.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite-State Morphology: Xerox Tools and Techniques*. CSLI Publications, Palo Alto.

Lauri Karttunen and Kenneth R. Beesley. 1992. *Two-Level Rule Compiler*. Technical Report. ISTL-1992-2. Xerox Palo Alto Research Center. Palo Alto, California.

Gilbert Lazard. 1992. *A Grammar of Contemporary Persian*. Mazda Publishers.

Shahrzad Mahootian. 1997. *Persian*.Routledge.

Karine Megerdoomian. 2000. Unification-Based Persian Morphology. In *Proceedings of CICLing 2000*. Alexander Gelbukh, ed. Centro de Investigación en Computación-IPN, Mexico.

Dariush Riazati.1997. *Computational Analysis of Persian Morphology*. MSc thesis, Department of Computer Science, RMIT.

Richard Sproat. 1992. *Morphology and Computation*.MIT Press, Cambridge, Massachusetts.

Arabic Script-based Languages deserve to be studied linguistically

Martin Kay
Stanford University

Arabic script-based languages are attracting increased attention for reasons that are regrettably far from their intrinsic linguistic interest. At the same time, statistical and corpus-based approaches to language processing are acquiring such dominance that it is becoming difficult for the advocates of other methods even to receive a hearing. I will argue that this is an alarming trend against which computational linguists, and especially those studying these languages, should resist with great determination. My argument for this position rests on the following observations:

1. Unless the role of quantum mechanics and chaos in the workings of ordinary language has been grossly underestimated, nothing about the subject is probabilistic in any fundamental sense.
2. The statistics are a surrogate for knowledge of the world and artificial intelligence and the performance of any system based on an approach that reduces this to numerical annotations on linguistic structures can only hope to reach a very low asymptote.
3. Thanks to Zipf's law, corpus annotation is subject to a severe law of diminishing returns to which the linguist's search for significant generalizations is not subject.
4. To the various levels of linguistic analysis and to the indefinite range of subjects and propositions that texts may treat, there correspond different notions of locality, each requiring its own statistical models.
5. Most importantly, most of the linguistic properties that must be considered for text processing are not emergent properties of the texts at all but crucially depend on *l'arbitraire du signe*, the arbitrary relation between a symbol and what it symbolizes.

An Unsupervised Approach for Bootstrapping Arabic Sense Tagging

Mona T. Diab
Stanford University
Stanford, CA 94305, USA
mdiab@stanford.edu

Abstract

To date, there are no WSD systems for Arabic. In this paper we present and evaluate a novel unsupervised approach, SALAAM, which exploits translational correspondences between words in a parallel Arabic English corpus to annotate Arabic text using an English WordNet taxonomy. We illustrate that our approach is highly accurate in $\leq 90.1\%$ of the evaluated data items based on Arabic native judgement ratings and annotations. Moreover, the obtained results are competitive with state-of-the-art unsupervised English WSD systems when evaluated on English data.

1 Introduction

Word Sense Disambiguation (WSD) is the process of resolving the meaning of a word unambiguously in a given natural language context. Within the scope of this paper, it is the process of marking text with an explicit set of sense tags or labels from some predefined tag set. It is well established that in order to obtain best quality sense annotations of words in running text, one needs a wide coverage lexicon and a trained lexicographer to annotate the words manually with their appropriate senses. Such a task is very tedious, expensive, and, by many standards, daunting to the people involved, even when all the required resources are available (Fellbaum et al., 2001). The problem becomes ever more challenging when dealing with a language with virtually no automated knowledge resources or tools. Like the majority of natural languages, the Arabic language happens to fall in this category of languages with minimal automated resources.

The focus of this paper is the sense disambiguation of Modern Standard Arabic which is the language used in formal speech and writing in the Arab world; Moreover, the script is shared with Urdu, Farsi, Dari and Pashtu. To our knowledge, there are no Arabic WSD systems reported in the literature.

Arabic is a Semitic language with rich templatic morphology. An Arabic word in text or speech may be composed of a stem, plus affixes and clitics. The affixes include inflectional markers for tense, gender, and/or number. The clitics include some (but not all) prepositions, conjunctions, determiners, possessive pronouns and pronouns. The stems consist of an underlying consonantal root and a template. The root could be anywhere from two to four consonants devoid of vocalization. Typically text in Modern Standard Arabic is written in the stem surface form with the various affixes. However, most Arabic dictionaries list the entries in terms of roots rather than surface forms.

In this paper, we present an approach, SALAAM (Sense Annotations Leveraging Alignments And Multilinguality), to bootstrap WSD for Arabic text presented in surface form. The approach of SALAAM is based on work by (Diab and Resnik, 2002) but it goes beyond it in the sense of extending the approach to the tagging of Arabic as a target language.(Diab, 2003) SALAAM uses cross-linguistic correspondences for characterizing word meanings in natural language. This idea is explored by several researchers, (Resnik and Yarowsky, 1998; Chugur et al., 2002; Ide, 2000; Dyvik, 1998). Basically, a word meaning or a word sense is quantifiable as much as it is uniquely translated in some language or set of languages. SALAAM is an empirical validation of this very notion of characterizing word meaning using cross-linguistic correspondences. Since automated lexical resources are virtually non-existent for Arabic, SALAAM leverages sense ambiguity resolution for Arabic off of existing English lexical resources and an Arabic English parallel corpus, thereby providing a bilingual solution to the WSD problem.

The paper is organized as follows: Section 2 describes the SALAAM system; Section 3 presents an evaluation of the approach followed by Section 4 which discusses the chosen sense inventory in relation to the Arabic data; We conclude with a summary and some final remarks in Section 6.

2 Approach

SALAAM exploits parallel corpora for sense annotation. The key intuition behind SALAAM is that when words in one language, *L1*, are translated into the same word in a second language, *L2*, then the *L1* words are semantically similar. For example, when the English — *L1* — words *bank, brokerage, mortgage-lender* translate into the Arabic — *L2* — word *bnk* (بنك) in a parallel corpus,[1] where the *bank* is polysemous, SALAAM discovers that the intended sense for the English word *bank* is the *financial institution* sense, not the *geological formation* sense, based on the fact that it is grouped with *brokerage* and *mortgage-lender*. Two fundamental observations are at the core of SALAAM:

- **Translation Distinction Observation (TDO)**

  Senses of ambiguous words in one language are often translated into distinct words in a second language.

  To exemplify **TDO**, we consider a sentence such as *I walked by the bank*, where the word *bank* is ambiguous with n senses. A translator may translate *bank* into *Dfp* (ضفه) corresponding to the *GEOLOGICAL FORMATION* sense or to *bnk* (بنك) corresponding to the *FINANCIAL INSTITUTION* sense depending on the surrounding context of the given sentence. Essentially, translation has distinctly differentiated two of the possible senses of *bank*.

- **Foregrounding Observation (FGO)**

  If two or more words are translated into the same word in a second language, then they often share some element of meaning.

  FGO may be expressed in quantifiable terms as follows: if several words $(w_1, w_2, \ldots, w_x)$ in *L1* are translated into the same word form in *L2*, then $(w_1, w_2, \ldots, w_x)$ share some element of meaning which brings the corresponding relevant senses for each of these words to the foreground. For example, if the word *Dfp* (ضفه), in Arabic, translates in some instances in a corpus to *shore* and other instances to *bank*, then *shore* and *bank* share some meaning component that is highlighted by the fact that the translator chooses the same Arabic word for their translation. The word *Dfp* (ضفه), in this case, is referring to the concept of *LAND BY WATER SIDE*, thereby making the corresponding senses in the English words more salient. It is important to note that the foregrounded senses of *bank* and *shore* are not necessarily identical, but they are quantifiably the closest senses to one another among the various senses of both words.

Given observations **TDO** and **FGO**, the crux of the SALAAM approach aims to quantifiably exploit the translator's implicit knowledge of sense representation cross-linguistically, in effect, reverse engineering a relevant part of the translation process.

SALAAM's algorithm is as follows:

- SALAAM expects a word aligned parallel corpus as input;
- *L1* words that translate into the same *L2* word are grouped into clusters;
- SALAAM identifies the appropriate senses for the words in those clusters based on the words senses' proximity in WordNet. The word sense proximity is measured in information theoretic terms based on an algorithm by Resnik (Resnik, 1999);
- A sense selection criterion is applied to choose the appropriate sense label or set of sense labels for each word in the cluster;
- The chosen sense tags for the words in the cluster are propagated back to their respective contexts in the parallel text. Simultaneously, SALAAM projects the propagated sense tags for *L1* words onto their *L2* corresponding translations.

The focus of this paper is on the last point in the SALAAM algorithm, namely, the sense projection phase onto the *L2* words in context. In this case, the *L2* words are Arabic and the sense inventory is the English WordNet taxonomy. Using SALAAM we annotate Arabic words with their meaning definitions from the English WordNet taxonomy. We justify the usage of an English inventory on both empirical and theoretical grounds. Empirically, there are no automated sense inventories for Arabic; Furthermore, to our knowledge the existing MRDs for Arabic are mostly root based which introduces another layer of ambiguity into Arabic processing

[1] We use the Buckwalter transliteration scheme for the Arabic words in this paper. http://www.ldc.org/aramorph

since Modern Standard Arabic text is rendered in a surface form relatively removed from the underlying root form. Theoretically, we subscribe to the premise that people share basic conceptual notions which are a consequence of shared human experience and perception regardless of their respective languages. This premise is supported by the fact that we have translations in the first place. Accordingly, basing the sense tagging of *L2* words with corresponding *L1* sense tags captures this very idea of shared meaning across languages and exploits it as a bridge to explicitly define and bootstrap sense tagging in *L2*, Arabic.

3 Evaluation

In order to formally evaluate SALAAM for Arabic WSD, there are several intermediary steps. SALAAM requires a token aligned parallel corpus as input and a sense inventory for one of the languages of the parallel corpus. For evaluation purposes, we need a manually annotated gold standard set.

3.1 Gold Standard Set

As mentioned above, there are no systems that perform Arabic WSD, therefore there exist no Arabic gold standard sets as such. Consequently, one needs to create a gold standard. Since SALAAM depends on parallel corpora, an English gold standard with projected sense tags onto corresponding Arabic words would serve as a good start. A desirable gold standard would be generic covering several domains, and would exist in translation to Arabic. Finding an appropriate English gold standard that satisfies both attributes is a challenge. One option is to create a gold standard based on an existing parallel corpus such as the Quran, the Bible or the UN proceedings. Such corpora are single domain corpora and/or their language is stylistic and distant from everyday Arabic; Moreover, the cost of creating a manual gold standard is daunting. Alternatively, the second option is to find an existing English gold standard that is diverse in its domain coverage and is clearly documented. Fortunately, the SENSEVAL2 exercises afford such sets.[2] SENSEVAL is a series of community-wide exercises that create a platform for researchers to evaluate their WSD systems on a myriad of languages using different techiques by constantly defining consistent standards and robust measures for WSD.

Accordingly, the gold standard set used here is the set of 671 Arabic words corresponding to the correctly sense annotated English nouns from the SENSEVAL2 English All Words Task. SALAAM achieved a precision of 64.5% and recall of 53% on the English test set for that task. SALAAM ranks as the best unsupervised system when compared to state-of-the-art WSD systems on the same English task. The English All Words task requires the WSD system to sense tag every content word in an English language text.

3.2 Token Aligned Parallel Corpora

The gold standard set corresponds to the test set in an unsupervised setting. Therefore the test set corpus is the SENSEVAL2 English All Words test corpus which comprises three articles from the Wall Street Journal discussing religious practice, medicine and education. The test corpus does not exist in Arabic. Due to the high expense of manually creating a parallel corpus, i.e. using human translators, we opt for automatic translation systems in a fashion similar to (Diab, 2000). To our knowledge there exist two off the shelf English Arabic Machine Translation (MT) systems: Tarjim and Almisbar.[3] We use both MT systems to translate the test corpus into Arabic. We merge the outputs of both in an attempt to achieve more variability in translation as an approximation to human quality translation. The merging process is based on the assumption that the MT systems rely on different sources of knowledge, different dictionaries in the least, in their translation process.

Fortunately, the MT systems produce sentence aligned parallel corpora.[4] However, SALAAM expects token aligned parallel corpora. There are several token alignment programs available. We use the GIZA++ package which is based on the IBM Statistical MT models.[5] Like most stochastic NLP applications, GIZA++ requires large amounts of data to produce reliable quality alignments. The test corpus is small comprising 242 lines only; Consequently, we augment the test corpus with several other corpora. The augmented corpora need to have similar attributes to the test corpus in genre and style. The chosen corpora and their relative sizes are listed in Table 1.

BC-SV1 is the Brown Corpus and SENSEVAL1 trial, training and test data. SV2-LS is the SENSEVAL2 English Lexical Sample trial, training and test data. WSJ is the Wall Street Journal. Finally SV2AW is SENSEVAL2 English All Words test corpus.

[2] http://www.senseval.org

[3] http://www.Tarjim.com, http://www.almisbar.com

[4] This is not a trivial problem with naturally occurring parallel corpora.

[5] http://www.isi.edu/och/GIZA++.html

Corpora	Lines	Tokens
BC-SV1	101841	2498405
SV2-LS	74552	1760522
WSJ	49679	1290297
SV2AW	**242**	**5815**
Total	*226314*	*5555039*

Table 1: Relative sizes of corpora used for evaluating SALAAM

The three augmenting corpora, BC-SV1, SV2LS and WSJ are translated into Arabic using both MT systems, AlMisbar and Tarjim. All the Arabic corpora are transliterated using the Buckwalter transliteration scheme and then tokenized. The corpora are finally token aligned using GIZA++. Figure 1 illustrates the first sentence of the SV2AW English test corpus with its translation into Arabic using AlMisbar MT system followed by its transliteration and tokenization, respectively.[6]

The art of change-ringing is peculiar to the English, and, like most English peculiarities, unintelligible to the rest of the world.

إن فن تغيير الدقاق خاص بالأنجليز، ومثل أكثر الخواص الأنجليزية، غير واضح إلى بقية العالم

{n fn tgyyr AldqAq xAS bAl{njlyz, wmvl Akvr AlxwAS Al{njlyzyp, gyr wADH Ila bqyp AlEAlm.

{n fn tgyyr Al dqAq xAS b Al {njlyz , w mvl Akvr Al xwAS Al {njlyzyp , gyr wADH Ila bqyp Al EAlm .

Figure 1: First sentence in test corpus SV2AW and its Arabic translation, transliteration and tokenization

3.3 Sense Inventory

The gold standard set is annotated using the WordNet taxonomy, WN1.7pre, for English. Like previous WordNet editions (Fellbaum, 1998), WN17pre is a computational semantic lexicon for English. It is rapidly becoming the community standard lexical resource for English since it is freely available for academic research. It is an enumerative lexicon in a Quillian style semantic network that combines the knowledge found in traditional dictionaries (Quillian, 1968). Words are represented as concepts, referred to as synsets, that are connected via different types of relations such as hyponymy, hypernymy, synonymy, meronymy, antonymy, etc. Words are represented as their synsets in the lexicon. For example, the word *bank* has 10 synsets in WN17pre corresponding to 10 different senses. The concepts are organized taxonomically in a hierarchical structure with the more abstract or broader concepts at the top of the tree and the specific concepts toward the bottom of the tree. For instance, the concept *FOOD* is the hypernym of the concept *FRUIT*, for instance.

Similar to previous WordNet taxonomies, WN17pre comprises four databases for the four major parts of speech in the English language: nouns, verbs, adjectives, and adverbs. The nouns database consists of 69K concepts and has a depth of 15 nodes. The nouns database is the richest of the 4 databases. Majority of concepts are connected via the IS-A identity relation. The focus of this paper is exclusively on nouns.[7]

3.4 Experiment and Metrics

We conducted two experiments.

3.4.1 Experiment 1

In the first experiment a native speaker of Arabic with near native proficiency in English is asked to pick the appropriate meaning definition of an Arabic word — given in its Arabic context sentence in which it appears in the corpus — from the list of WN1.7pre definitions. They are allowed to pick more than one definition for each item. Or alternatively, the annotator has the option to choose *NONE* where none of the definitions is appropriate for the Arabic word given the Arabic context sentence; Or *MISALIGNMENT* where the Arabic word is not a translation of the English word whose meaning definitions appear in the list that follows, or it is simply a misalignment. The results from this experiment are illustrated in Table 2.

Category	Num. of items	%
Agreement	605	90.1
Disagreement	21	3.1
None	1	0.14
Misalignment	44	6.55

Table 2: Human Annotator agreement scores with SALAAM automatic annotations.

It is worth noting the high agreement rate between the annotator and the SALAAM annotations

[6] All the Arabic sentences in this paper are output from one of the MT systems used.

[7] SALAAM, however, has no inherent restriction on part of speech.

which exceed 90%. The only case that is considered a "NONE" category is for the word *bit* which is translated as the past tense of *to bite* as عض. It should have been translated as قطعة meaning a *morsel/piece*.

3.4.2 Experiment 2

In this experiment, the Arabic words annotated with English WN1.7pre tags are judged on a five point scale metric by three native speakers of Arabic with near native proficiency in English. The experiment is run in a form format on the web. The raters are asked to judge the accurateness of the chosen sense definition from a list of definitions associated with the translation of the Arabic word. The Arabic words are given to the raters in their respective context sentences. Therefore the task of the rater is to judge the appropriateness of the chosen English sense definition for the Arabic word given its context. S/he is required to pick a rating from a drop down menu for each of the data items. The five point scale is as follows:

- **Accurate**: This choice indicates that the chosen sense definition is an appropriate meaning definition of the Arabic word.
- **Approximate**: This choice indicates that the chosen sense definition is a good meaning definition for the Arabic word given the context yet there exists on the list of possible definitions a more appropriate sense definition.
- **Misalignment**: This choice indicates that the Arabic word is not a translation of the English word due to a misalignment or the word being rendered in English in the Arabic sentence, i.e. the English word was not translated by either of the Arabic MT systems.
- **None**: This choice indicates that none of the sense definitions listed is an appropriate sense definition for the Arabic word.
- **Wrong**: This choice indicates that the chosen sense definition is the incorrect meaning definition for the Arabic word given its context.

3.5 Results

Table 3 illustrates the obtained results from the three raters.

The inter-rater agreement is at a high 96%. They all deemed on average more than 90% of the data items to be accurately tagged by SALAAM. The most variation seemed to be in assessing the APPROXIMATE category with Rater 1, R1, rating 19 items as APPROXIMATE and R2 rating 10 items as APPROXIMATE and R3 rating 14 data items as APPROXIMATE.

Type	R1	R2	R3
Accurate	90.3	90.4	91.4
Approximate	2.8	2	1.5
Misalignment	5.6	5.9	5.9
None	0	0	0
Wrong	1.2	1.3	1.2

Table 3: Rater judgments on the Arabic WSD using meaning definitions from the English WN1.7pre

An example of a data item that is deemed APPROXIMATE by the three raters is for the word *AltjmE* (التجمع) in the following sentence:

تدق فرقة جديدة كليا كل يوم في تورنجتون العظيمة، عدة من أعضاء التجمع

transliterated as

tdq frqp jdydp klyA kl ywm fy twrnjtwn AlEZymp, edp mn AED' AltjmE

which means

In Great Torington, a brand new band plays everyday comprising members of the congregation

The word *AltjmE* (التجمع) is a translation of *congregation* which has the following sense definitions in WN1.7pre:

- `congregation: an assemblage of people or animals or things collected together; "a congregation of children pleaded for his autograph"; "a great congregation of birds flew over"`
- `congregation, fold, faithful: a group of people who adhere to a common faith and habitually attend a given church`
- `congregation, congregating: the act of congregating`

SALAAM favors the last meaning definition for *congregation*.

An example of a MISALIGNMENT is illustrated in the following sentence:

القولون والرئه وسرطان الثدى أكثر الأشكال القاتله للمرض

transliterated as

Alqwlwn wAlr'p wsrTAn Alvdy Akvr AlA$kAl AlqAtlp llmrD...

which is a translation of

Cancer of the Colon, Breast and Lungs are the most deadly forms of the disease...

The words *srTAn* (سرطان), meaning *cancer*, and *lungs* were aligned leading to tagging the Arabic word with the sense tag for the English word *lungs*. Finally, the following is an example of a WRONG data item as deemed by the three raters. The definition for the word *Alywm* (اليوم) in the following sentence:

يعيش الأخرون اليوم فى مكان آخر

transliterated as

yEy$ AlAxrwn Alywm fy mkAn Axr...

which means

The others live today in a different place...

where the word equivalent to *today* is the target word with the following sense definitions:

- `today: the day that includes the present moment (as opposed to yesterday or tomorrow); "Today is beautiful"; "did you see today's newspaper?"`
- `today: the present time or age; "the world of today"; "today we have computers"`

SALAAM chooses the first meaning definition while the raters seem to favor the second.

None of the raters seemed to find data items that had no corresponding meaning definition in the given list of English meaning definitions. It is interesting to note that the single item considered a "NONE" category in experiment 1 was considered a misalignment by the three raters.

If we calculate the average precision of the evaluated sense tagged Arabic words based on the total tagged English nouns of 1071 nouns in this test set, we obtain an absolute precision of 56.9% for Arabic sense tagging. It is worth noting that the average precision on the SENSEVAL2 English All Words Task for any of the unsupervised systems is in the lower 50% range.

4 General Discussion

It is worth noting the high agreement level between the rating judgments of the three raters in experiment 2 and the human manual annotations of experiment 1. The obtained results are very encouraging indeed but it makes the implicit assumption that the English WordNet taxonomy is sufficient for meaning representation of the Arabic words used in this text. In this section, we discuss the quality of WN1.7pre as an appropriate sense inventory for the Arabic task.

With that intent in mind, we evaluate the 600 word instances of Arabic that are deemed correctly tagged using the English WN17pre.[8] We investigate three different aspects of the Arabic English correspondence: Arabic and English words are equivalent; Arabic words correspond to specific English senses; And English words do not sufficiently correspond to all possible senses for the Arabic word. The three aspects are discussed in detail below.

- **Arabic and English words are equivalent**

  We observe that a majority of the ambiguous words in Arabic are also ambiguous in English in this test set; they preserve ambiguity in the same manner. In Arabic, 422 word tokens corresponding to 190 word types, are at the closest granularity level with their English correspondent;[9] For instance, all the senses of *care* apply to its Arabic translation *EnAyA* (عنايه); the sense definitions are listed as follows:

  - `care, attention, aid, tending: the work of caring for or attending to someone or something; "no medical care was required"; "the old car needed constant attention"`
  - `caution, precaution, care, forethought: judiciousness in avoiding harm or danger; "he exercised caution in opening the door"; "he handled the vase with care"`

[8] The overlapping number of Arabic words rated ACCURATE by the three annotators of experiment 1 and those accurate items from experiment 1.

[9] This means that all the English senses listed for WN17pre are also senses for the Arabic word.

- concern, care, fear: an anxious feeling; "care had aged him"; "they hushed it up out of fear of public reaction"
- care: a cause for feeling concern; "his major care was the illness of his wife"
- care, charge, tutelage, guardianship: attention and management implying responsibility for safety; "he is under the care of a physician"
- care, maintenance, upkeep: activity involved in maintaining something in good working order; "he wrote the manual on car care"

It is worth noting that the cases where ambiguity is preserved in English and Arabic are all cases where the polysemous word exhibits regular polysemy and/or metonymy. The instances where homonymy is preserved are borrowings from English. Metonymy is more pragmatic than regular polysemy (Cruse, 1986); for example, *tea* in English has the following metonymic sense from WN1.7pre:

- a reception or party at which tea is served; "we met at the Dean's tea for newcomers"

This sense of *tea* does not have a correspondent in the Arabic *$Ay* (شاى). Yet, the English *lamb* has the metonymic sense of *MEAT* which exists in Arabic. Researchers building EuroWordNet have been able to devise a number of consistent metonymic relations that hold cross linguistically such as *fabric/material*, *animal/food*, *building/organization* (Vossen et al., 1999; Wim Peters and Wilks, 2001). In general, in Arabic, these defined classes seem to hold, however, the specific case of *tea* and *party* does not exist. In Arabic, the English sense would be expressed as a compound *tea party* or *Hflp $Ay* (حفلة شاى).

- **Arabic word equivalent to specific English sense(s)**

In this evaluation set, there are 138 instances where the Arabic word is equivalent to a subsense(s) of the corresponding English word. The 138 instances correspond to 87 word types. An example is illustrated by the noun *ceiling* in English.

- ceiling: the overhead upper surface of a room; "he hated painting the ceiling"
- ceiling: (meteorology) altitude of the lowest layer of clouds
- ceiling, cap: an upper limit on what is allowed: "they established a cap for prices"
- ceiling: maximum altitude at which a plane can fly (under specified conditions)

The correct sense tag assigned by SALAAM to *ceiling* in English is the first sense, which is correct for the Arabic translation *sqf* (سقف). Yet, the other 3 senses are not correct translations for the Arabic word. For instance, the second sense definition would be translated as {*rtfAE* (إرتفاع) and the last sense definition would be rendered in Arabic as *Elw* (علو). This phenomenon of Arabic words corresponding to specific English senses and not others is particularly dominant where the English word is homonymic. By definition, homonymy is when two independent concepts share the same orthographic form, in most cases, by historical accident. Homonymy is typically preserved between languages that share common origins or in cases of cross-linguistic borrowings. Owing to the family distance between English and Arabic, polysemous words in Arabic rarely preserve homonymy.

- **English word equivalent to specific Arabic sense**

40 instances, corresponding to 20 type words in Arabic, are manually classified as more generic concepts than their English counterparts. For these cases, the Arabic word is more polysemous than the English word. For example, the English noun *experience* possesses three senses in WN17pre as listed below.

- `experience: the accumulation of knowledge or skill that results from direct participation in events or activities; "a man of experience"; "experience is the best teacher"`
- `experience: the content of direct observation or participation in an event; "he had a religious experience"; "he recalled the experience vividly"`
- `experience: an event as apprehended; "a surprising experience"; "that painful experience certainly got our attention"`

All three senses are appropriate meanings of the equivalent Arabic word *tjrbp* (تجربة) but they do not include the *SCIENTIFIC EXPERIMENT* sense covered by the Arabic word.

From the above points, we find that 63.9% of the ambiguous Arabic word types evaluated are conceptually equivalent to their ambiguous English translations. This finding is consistent with the observation of EuroWordNet builders. Vossen, Peters, and Gonzalo (1999) find that approximately 44-55% of ambiguous words in Spanish, Dutch and Italian have relatively high overlaps in concept and the sense packaging of polysemous words (Vossen et al., 1999). 29.3% of the ambiguous Arabic words correspond to specific senses of their English translations and 6.7% of the Arabic words are more generic than their English correspondents.

5 Acknowledgements

I would like to thank Philip Resnik and Daniel Jurafsky for their insightful comments. I would like to thank two annonymous reviewers for their detailed comments. This work is supported, in part, by NSF Award #IIS-0325646.

6 Conclusions

We presented, SALAAM, a method for bootstrapping the sense disambiguation process for Arabic texts using an existing English sense inventory leveraging translational correspondence between Arabic and English. SALAAM achieves an absolute precision of 56.9% on the task for Arabic. Of the 673 correctly tagged English tokens for the SENSEVAL2 English All Words Task, approximately 90% of the Arabic data is deemed correctly tagged by 3 native speakers of Arabic. Therefore, SALAAM is validated as a very good first approach to Arabic WSD. Moreover, we perform a preliminary investigation with very promising results into the quality of the English sense inventory, WN1.7pre, as an approximation to an Arabic sense inventory.

References

Irina Chugur, Julio Gonzalo, and Felisa Verdejo. 2002. Polysemy and sense proximity in the senseval-2 test suite. In *Proceedings of Word Sense Diasmbiguation: Recent Successes and Future Directions*, University of Pennsylvania, Pennsylvania, July.

D. Cruse. 1986. *Lexical Semantics*. Cambridge University Press.

Mona Diab and Philip Resnik. 2002. Word sense tagging using parallel corpora. In *Proceedings of 40th ACL Conference*, Pennsylvania, USA.

Mona Diab. 2000. An unsupervised method for multilingual word sense tagging using parallel corpora: A preliminary investigation. In *SIGLEX2000: Word Senses and Multi-linguality*, Hong Kong, October.

Mona Diab. 2003. Word sense disambiguation within a multilingual framework. In *PhD Thesis*, University of Maryland, College Park.

Helge Dyvik. 1998. Translations as semantic mirrors.

Christiane Fellbaum, Martha Palmer, Hoa Trang Dang, Lauren Delfs, and Susanne Wolff. 2001. Manual and Automatic Semantic Annotation with WordNet. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources: Applications, Customizations*, Carnegie Mellon University, Pittsburg, PA.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press. http://www.cogsci.princeton.edu/~wn [2000, September 7].

Nancy Ide. 2000. Cross-lingual sense discrimination: Can it work? *Computers and the Humanities*, 34:223–34.

M.R. Quillian. 1968. Semantic Memory. In M. Minsky, editor, *Semantic Information Processing*. The MIT Press, Cambridge, MA.

Philip Resnik and David Yarowsky. 1998. Distinguishing Systems and Distinguishing Senses: New Evaluation Methods for Word Sense Disambiguation. *Natural Language Engineering*, 1(1):1–25.

Philip Resnik. 1999. Disambiguating Noun Groupings with Respect to WordNet Senses. In S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann, and D. Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*, pages 77–98. Kluwer Academic, Dordrecht.

P. Vossen, W. Peters, and J. Gonzalo. 1999. Towards a Universal Index of Meaning. pages 1–24.

Louise Guthrie Wim Peters and Yorick Wilks. 2001. Cross-linguistic discovery of semantic regularity.

Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm

Mohamed EL KOURDI **Amine BENSAID**♠ **Tajje-eddine RACHIDI**

School of Science & Engineering
Alakhawayn University
P.O. Box 104, Ifrane 53000, Morocco
[M.Elkourdi, A.Bensaid, T.Rachidi]@alakhawayn.ma
♠Corresponding Author

Abstract

This paper deals with automatic classification of Arabic web documents. Such a classification is very useful for affording directory search functionality, which has been used by many web portals and search engines to cope with an ever-increasing number of documents on the web. In this paper, Naive Bayes (NB) which is a statistical machine learning algorithm, is used to classify non-vocalized Arabic web documents (after their words have been transformed to the corresponding canonical form, i.e., roots) to one of five pre-defined categories. Cross validation experiments are used to evaluate the NB categorizer. The data set used during these experiments consists of 300 web documents per category. The results of cross validation in the leave-one-out experiment show that, using 2,000 terms/roots, the categorization accuracy varies from one category to another with an average accuracy over all categories of 68.78 %. Furthermore, the best categorization performance by category during cross validation experiments goes up to 92.8%. Further tests carried out on a manually collected evaluation set which consists of 10 documents from each of the 5 categories, show that the overall classification accuracy achieved over all categories is 62%, and that the best result by category reaches 90%.

Keywords: Naïve Bayes, Arabic document categorization, cross validation, TF-IDF.

1 Introduction

With the explosive growth of text documents on the web, relevant information retrieval has become a crucial task to satisfy the needs of different end users. To this end, automatic text categorization has emerged as a way to cope with such a problem. Automatic text (or document) categorization attempts to replace and save human effort required in performing manual categorization. It consists of assigning and labeling documents using a set of pre-defined categories based on document contents. As such, one of the primary objectives of automatic text categorization has been the enhancement and the support of information retrieval tasks to tackle problems, such as information filtering and routing, clustering of related documents, and the classification of documents into pre-specified subject themes. Automatic text categorization has been used in search engines, digital library systems, and document management systems (Yang, 1999). Such applications have included electronic email filtering, newsgroups classification, and survey data grouping. Barq for instance uses automatic categorization to provide similar documents feature (Rachidi et al., 2003). In this paper, NB which is a statistical machine learning algorithm is used to learn to classify non-vocalized[1] Arabic web text documents.

This paper is organized as follows. Section 2, briefly describe related works in the area of automatic text categorization. Section 3 describes the preprocessing undergone by documents for the purpose of categorization; it describes in particular the preprocessing specific to the Arabic language. In section 4 Naïve Bayes (NB), the learning algorithm used in this paper for document categorization is presented. Section 5 outlines the experimental setting, as well as the experiments carried out to evaluate the performance of the NB classifier. It also gives the numerical results with their analysis and interpretation. Section 6 summarizes the work and suggests some ideas for future works.

2 Related Works

Many machine learning algorithms have been applied for many years to text categorization. They

[1] Most modern Arabic writing (web, novels, articles) are written without vowels.

include decision tree learning and Bayesian learning, nearest neighbor learning, and artificial neural networks, early such works may be found in (Lewis and Ringnette, 1994), (Creecy and Masand, 1992) and (Wiene and Pedersen, 1995), respectively.

The bulk of the text categorization work has been devoted to cope with automatic categorization of English and Latin character documents. For example, (Fang et al., 2001) discusses the evaluation of two different text categorization strategies with several variations of their feature spaces. A good study comparing document categorization algorithms can be found in (Yang and Liu, 1999). More recently, (Sebastiani, 2002) has performed a good survey of document categorization; recent works can also be found in (Joachims, 2002), (Crammer and Singer, 2003), and (Lewis et al., 2004).

Concerning Arabic, one automatic categorizer has been reported to have been put under operational use to classify Arabic documents; it is referred to as "Sakhr's categorizer" (Sakhr, 2004). Unfortunately, there is no technical documentation or specification concerning this Arabic categorizer. Sakhr's marketing literature claims that this categorizer is based on Arabic morphology and some research that has been carried out on natural language processing.

The present work evaluates the performance on Arabic documents of the Naïve Bayes algorithm (NB) - one of the simplest algorithms applied to English document categorization (Mitchell, 1997). The aim of this work is to gain some insight as to whether Arabic document categorization (using NB) is sensitive to the root extraction algorithm used or to different data sets. This work is a continuation of that initiated in (Yahyaoui, 2001), which reports an overall NB classification correctness of 75.6%, in cross validation experiments, on a data set that consists of 100 documents for each of 12 categories (the data set is collected from different Arabic portals). A 50% overall classification accuracy is also reported when testing with a separately collected evaluation set (3 documents for each of the 12 categories). The present work expands the work in (Yahyaoui, 2001) by experimenting with the use of a better root extraction algorithm (El Kourdi, 2004) for document preprocessing, and using a different data set, collected from the largest Arabic site on the web: aljazeera.net.

3 Preprocessing of document

Prior to applying document categorization techniques to an Arabic document, the latter is typically preprocessed: it is parsed, in order to remove stopwords (these are conjunction and disjunction words etc.). In addition, at this stage in this work, vowels are stripped from the full text representation when the document is (fully or partially) voweled/vocalized. Then roots are extracted for words in the document.

In Arabic, however, the use of stems will not yield satisfactory categorization. This is mainly due to the fact that Arabic is a non-concatenative language (Al-Shalabi and Evens, 1998), and that the stem/infix obtained by suppression of infix and prefix add-ons is not the same for words derived from the same origin called the root. The infix form (or stem) needs further to be processed in order to obtain the root. This processing is not straightforward: it necessitates expert knowledge in Arabic language word morphology (Al-Shalabi and Evens, 1998). As an example, two close roots (i.e., roots made of the same letters), but semantically different, can yield the same infix form thus creating ambiguity.

The root extraction process is concerned with the transformation of all Arabic word derivatives to their single common root or canonical form. This process is very useful in terms of reducing and compressing the indexing structure, and in taking advantage of the semantic/conceptual relationships between the different forms of the same root. In this work, we use the Arabic root extraction technique in (El Kourdi, 2004). It compares favorably to other stemming or root extraction algorithms (Yates and Neto, 1999; Al-Shalabi and Evens, 1998; and Houmame, 1999), with a performance of over 97% for extracting the correct root in web documents, and it addresses the challenge of the Arabic broken plural and hollow verbs. In the remainder of this paper, we will use the term "root" and "term" interchangeably to refer to canonical forms obtained through this root extraction process.

4 NB for document categorization

4.1 The classifier module

The classifier module is considered to be the core component of the document categorizer. It is responsible for classifying given Arabic documents to their target class. This is performed using the Naive Bayes (NB) algorithm. The NB classifier

computes a posteriori probabilities of classes, using estimates obtained from a training set of labeled documents. When an unlabeled document is presented, the a posteriori probability is computed for each class using (1) in Figure 1; and the unlabeled document is then assigned to the class with the largest a posteriori probability.

A posteriori probability computation

Let D be a document represented as a set of finite terms $D=\{w_1, w_2,..., w_3\}$.

Let C be the number of target classes.

Let $docs_i$ *be the number of documents in category* C_i *and |Examples| be the number of documents in the training set of labeled documents.*

Let n be the total number of distinct stems in C_i

Let N_k *be the number of times* w_k *occurs in* C_i

Then the a posteriori probability as given by Bayes theorem is:

$$P(C_i|D)=[P(C_i)*P(D|\ C_i)]/P(D).\ i=1,2,...C \quad \textbf{(1)}$$

When comparing a posteriori probabilities for the same document D, P(D) is the same for all categories and will not affect the comparison.

The other quantities in (1) are estimated from the training set using NB learning (see Figure 2).

The assigned class AC(D) to document D is the class with largest a posteriori probability given by (1):

$$AC(D)=argmax_{Ci}\ \{\ P(C_i|D).\ i=1,2,...C\}$$

Figure 1. A posteriori probability reduction.

4.2 The learning module

The main task of the learning module is to learn from a set of labeled documents with predefined categories in order to allow the categorizer to classify the newly encountered documents *D* and to assign them to each of the predefined target categories C_i. This module is based on the NB learning algorithm given in Figure 2. The learning module is one way of estimating the needed quantities in (1) by learning from a training set of documents.

NB learning algorithm

Let D be a document represented as a set of finite terms/roots $D=\{w_1, w_2,..., w_n\}$.

Let $docs_i$ *be the number of documents in category* C_i *, and |Examples| be the number of documents in the training set of labeled documents.*

Step 1: collect the vocabulary, which is defined as the set of distinct words in the whole training set

Step2: For each category C_i *do the following*

Compute $P(C_j) = |\ docs_j\ |/|Examples|$ ***(2)***

where $docs_j$ *is the number of training documents for the category is* C_j.

For each root w_k in Vocabulary

Compute $P(w_k/C_j)= (N_{k,j} +1)/(\ n_j +|\ Text_j\ |)$ ***(3)***

where $N_{k,j}$ *is the number of times* w_k *occurs in* C_j, n_j *is the total number of distinct terms in all training documents labeled* C_j, *and* $Text_j$ *is a single documents generated by concatenating all the training documents for category* C_j.

Equation (2) and (3) make use of the following two assumptions:

1) Assuming that the order of the words in a document does not affect the classification of the document:

$$P(D|C_j)=P(\{w_1, w_2,..., w_n\}|C_j) \quad \textbf{(4)}$$

2) Assuming that the occurrence of each word is independent of the occurrence of other words in the document then:

$$P(w_1,...,w_n|C_j)=P(w_1|C_j)*P(w_2|C_j)*...*P(w_n|C_j) \quad \textbf{(5)}$$

Figure 2. The Naïve Bayes (supervised) learning algorithm for document categorization

The m-estimate method (with m equal to the size of word vocabulary) (Cestink, 1990) is used to compute the probability terms and handle zero count probabilities (smoothing). Equation (3) gives an estimate for $P(w_k/C_j)$.

Various assumptions are needed in order to simplify Equation (1), whose computations are otherwise expensive. These assumptions are applied in Figure 2 to obtain the needed quantities

for the class-conditional probabilities (Equations (4) and (5)). These assumptions are:

1. The probability of encountering a specific word within a document is the same regardless the word position. In other words, $P(w_i=w|C_j)= P(w_m= w|C_j)$ for every i, j, and m where i and m are different possible positions of the same word within the document. This assumption allows representing a document as a bag of word (Equation (4) in Figure 2).

2. The probability of occurrence of a word is independent of the occurrence of other words in the same document. This is reflected in Equation (5): $P(w_1,...,w_n|C_j)=P(w_1|C_j)*P(w_2|C_j)*...*P(w_n|C_j)$. It is in fact a naïve assumption, but it significantly reduces computation costs, since the number of probabilities that should be computed is decreased. Even though this assumption does not hold in reality, NB performs surprisingly well for text classification (Mitchell, 1997).

5 Experiments and results

For classification problems, it is customary to measure a classifier's performance in terms of classification error rate. A data set of documents is used with known category/class label $L(D_k)$ for each document D_k. The set is split into two subsets: a training set and a testing set. The trained classifier is used to assign a class $AC(D_k)$ using Equation (3) to each document (D_k) in the test set, as if its true class label were not known. If $AC(D_k)$ matches $L(D_k)$, the classification is considered correct; otherwise, it is counted as an error:

$$\text{Error}_{ik}=\begin{cases} 1 & \text{iff } L(D_k)=C_i, \text{ and } AC(D_k)\neq C_i \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(6)}$$

For a given class, the error rate is computed as the ratio of the number of errors made on the whole test set of unlabeled documents (X^u) to the cardinality $|X^u|$ of this set. For a given class C_i, the error rate is computed as:

$$\text{ClassError}_i = \sum_{k=1}^{|X^u|} \text{Error}_{ik} \; / \; |X^u| \quad \textbf{(7)}$$

In order to measure the performance of the NB algorithm on Arabic document classification, we conducted several experiments: we performed cross validation using the original space (using all the words in the documents), cross validation experiments based on feature selection (using a subset of terms/roots only), and experiments based on an independently constructed evaluation set. The following paragraphs describe the data set used, and the experiments.

5.1 The data set

We have collected 300 web documents for each of five categories from the website www.aljazeera.net, which is the website of Aljazeera (the Qatari television news channel in Arabic). This site contains over seven million (7,000,000) documents corresponding to the programs broadcast on the television channel; it is arguably the most visited Arabic web site. Aljazeera.net presents documents in (manually constructed) categories. The five (5) categories used for this work are: sports, business, culture and art, science, and health.

5.2 Cross validation

In cross validation, a fixed number of documents is reserved for testing (as if they were unlabeled documents) and the remainder are used for training (as labeled documents). Several such partitions of the data set are constructed, by making random splits of the data set. NB's performance is evaluated several times, using the different random partitions. Then the error statistics are aggregated. The steps of the cross validation experiments are delineated in Figure 3 next:

Cross validation steps

Let X be the entire data seto f N=1500 documents

c =5 is the number of different categories

$E_{r,i}$ will store the error rate for category i during trial r.

1) Fix the size s of the training set for (s=N/3, N/2, 2N/3, or N-1) to perform 1/3-2/3, 50/50, 2/3-1/3 or leave-one-out cross validation.

2) Set the number of trials T. If s=N-1, fix the number of trials T=N; else, T=40.

3) For trial r=1 to T

3.1 Select randomly s documents from X as labeled documents into training set X_r^l.

3.2 Store the remaining documents (X- X_r^l) as unlabeled documents into X_r^u (as if they were unlabeled).

3.3 Train NB using X_r^l. (Compute Equation (2) and Equation (4))

3.4 Use trained NB to compute the class of each element in X_r^u using Equation (4)

3.5 Compute error rate $E_{r,i}$, on X_r^u for each category (i=1,2...,c) using Equation (7):

$$E_{r,i} = \sum_{k=1}^{|X^u|} \text{Error}_{ik} / |X^u| \quad i=1,2,\ldots,c$$

Next r (return to step 3).

4.1 Compute the average error rate for each class over all trials:

$$AvgError_{i,s} = \sum_{r=1}^{T} E_{r,i} / T \quad i=1,2,\ldots,c$$

4.2 Compute the maximum error rate for each class over all trials:

$$MaxError_{i,s} = \text{Max}_{r=1,2...,T} \{E_{r,i}\} \quad i=1,2,\ldots,c$$

4.3 Get the minimum error rate for each class over all trials:

$$MinError_{i,s} = \text{Min}_{r=1,2...,T} \{E_{r,i}\} \quad i=1,2,\ldots,c.$$

Next s (return to step 1)

Figure 3. Cross validation experiments.

5.2.1. Experiments without feature extraction

In these experiments, each document in data set X is represented by all word roots in the document. The cross validation experiments described in Figure 3, is conducted. Table 1 reports the error rates obtained over all categories during the cross validation experiments. The smallest error rate is obtained in the leave-one-out experiment (as illustrated in Table 1). Table 2, Table 3, Table 4, and Table 5 represent, respectively, the confusion matrices of the cross validation experiments. The percentages reported in an entry of a confusion matrix correspond to the percentage of documents that are known to actually belong to the category given by the row header of the matrix, but that are assigned by NB to the category given by the column header.

	Cross-validation Experiments			
	1/3-2/3	1/2-1/2	2/3-1/3	Leave-one-out
Avg	67%	55%	46%	32.1%
Max	69.9%	56.5%	49%	100%
Min	62%	48.1%	42%	0%

Error Rate

Table 1. The error rates of NB over all categories in cross validation experiments (with feature extraction)

Category	Health	Business	Culture	Science	Sport
Health	22%	27%	3%	8%	40%
Business	7%	39%	10%	18%	26%
Culture	13%	18%	27%	7%	35%
Science	14%	15%	8%	30%	33%
Sport	16%	12%	17%	8%	47%

Table 2. Confusion Matrix results for cross validation, with no feature extraction (1/3-2/3).

Category	health	Business	Culture	Science	Sport
Health	32%	22.5%	3.2%	8%	34.3 %
Business	8.2%	50%	10.7%	13.3%	17.8 %
Culture	8%	20%	39%	3%	30%
Science	16%	9.8%	7.2%	46%	21%
Sport	12%	8%	16%	4%	60%

Table 3. Confusion Matrix results for cross validation, with no feature extraction (1/2-1/2).

Category	Health	Business	Culture	Science	Sport
Health	46%	12%	6%	8%	28%
Business	4.8%	63%	7%	9.2%	16%
Culture	7.1%	16.8%	42%	6.1%	28%
Science	8.1%	10.8%	9.1%	46%	26%
Sport	7.2%	5%	6.8%	5%	76%

Table 4. Confusion Matrix results for cross validation, with no feature extraction (2/3-1/3).

Category name	Health	Business	Culture	Science	Sport
Health	58.0%	13%	4%	3.7%	21.3%
Business	4.6%	**73.5%**	5.3%	4.6%	12%
Culture	2.3%	10%	57.0%	0.7%	30%
Science	13.3%	5.3%	2.3%	59.1%	20%
Sport	2.0%	1.3%	3.6%	1.3%	**91.8%**

Table 5. Confusion Matrix results for cross validation, with no feature extraction (Leave-one-out)

The diagonals in tables 2-5 indicate higher classification performance for categories: Sport and Business than for the categories: Culture, Science, and health. Moreover, the leave-one-out experiment yields the best result by category as illustrated in Table 5 compared to the error rates reported in tables 2-4. Tables 2-5 revealed that error rates by

category decrease from experiment to experiment. In other words, the error rates recorded in 1/3-2/3 experiment are higher than those in 1/2-1/2 experiment, those in 1/2-1/2 experiment are higher than those in 2/3-1/3 experiment, and those obtained in the 2/3-1/3 experiment are higher than those in the leave-one-out experiment. Thus, larger training sets yield higher accuracy when all the data set terms are used.

When investigating some of the misclassifications/confusions made by NB, we have noticed that misclassified documents, in fact, contain large number of words that are representative of other categories. In other words, documents that are known to belong to a category contain numerous words that have higher frequency in other categories. Therefore, these words have higher influence on the prediction that will be made by the classifier. For instance, the confusion matrix in Table 5 shows that 30% of Culture documents have been misclassified in the Sports category. The misclassified documents contain words that are more frequent in the Sports category such as جائزة (Arabic for prize and for trophy), بطل (Arabic for champion and for lead character), and تسجيل (Arabic for scoring and for recording).

5.2.2. Cross-validation, using feature selection

Feature selection techniques have been widely used in information retrieval as a means for coping with the large number of words in a document; a selection is made to keep only the more relevant words. Various feature selection techniques have been used in automatic text categorization; they include document frequency (DF), information gain (IG) (Tzeras and Hartman, 1993), minimum description length principal (Lang, 1995), and the χ^2 statistic. (Yang and Pedersen, 1997) has found strong correlations between DF, IG and the χ^2 statistic for a term. On the other hand, (Rogati and Yang, 2002) reports the χ^2 to produce best performance. In this paper, we use TF-IDF (a kind of augmented DF) as a feature selection criterion, in order to ensure results are comparable with those in (Yahyaoui, 2001).

TF-IDF (term frequency-inverse document frequency) is one of the widely used feature selection techniques in information retrieval (Yates and Neto, 1999). Specifically, it is used as a metric for measuring the importance of a word in a document within a collection, so as to improve the recall and the precision of the retrieved documents. While the TF measurement concerns the importance of a term in a given document, IDF seeks to measure the relative importance of a term in a collection of documents. The importance of each term is assumed to be inversely proportional to the number of documents that contain that term. TF is given by $TF_{D,t}$, and it denotes frequency of term t in document D. IDF is given by $IDF_t = \log(N/df_t)$, where N is the number of documents in the collection, and df_t is the number of documents containing the term t. (Salton and Yang, 1973) proposed the combination of TF and IDF as weighting schemes, and it has been shown that their product gave better performance. Thus, the weight of each term/root in a document is given by $w_{D,t} = TF_{D,t} * IDF_t$.

We have conducted five cross validation experiments based on TF-IDF. Experiments are based on selecting, in turn, 50, 100, 500, 1000, and 2000 terms that best represent the predefined 5 categories. We have repeated the experiments in Figure 3 for each number of terms. A summary of the results is presented in Table 6. The performance levels obtained are comparable to those obtained without feature selection. Figure 4 plots average categorization error rates versus the number of terms used for different trials.

Experiments / #terms/roots	1/3-2/3	1/2-1/2	2/3-1/3	Leave-one-out
50	73.2(69.9,77.5)	64.8(60.5,72.8)	75.8(69.8,82.3)	36.9(0,100)
100	72.4(69.7,77)	62.5(58.9,66.7)	69.4(46.8,82.5)	33.7(0,100)
500	71.1(68.9,74.5)	60.2(57.9,64.2)	63.9(49.5,68.2)	33.16(0,100)
1000	69.7(64.4,73.2)	57.1(48.2,59.2)	44.9(42.8,48.7)	32.18(0,100)
2000	66.1(59.1,69)	53.9(44.9,59)	43.3(40.1,47.5)	31.22(0,100)
5000	67(62.6,69.9)	55(49.1,56.5)	46(42,49)	32.1(0,100)

Table 6. The overall error rate of NB in cross validation experiments using feature selection, in format: Avg(Min, Max)

Category	NB accuracy
Health	50%
Business	70%
Culture	40%
Science	60%
Sport	90%

Table 7. Classification accuracy on the evaluation set using Leave-one-out and TF-IDF with 2,000 roots/terms

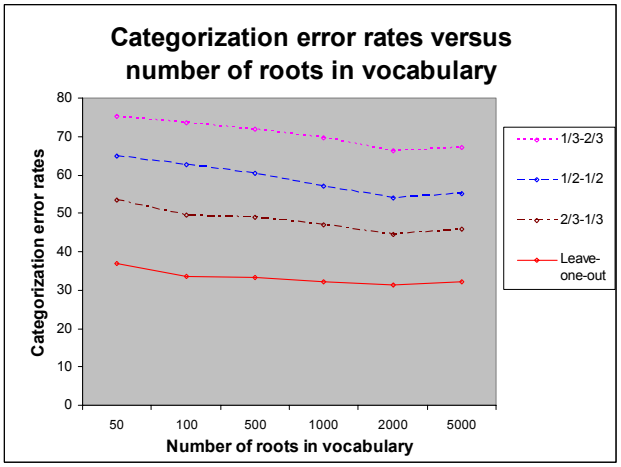


Figure 4. Categorization error rates versus number of terms.

5.3 Experiments using an evaluation set

Cross validation has been used to determine the average performance of NB for Arabic text categorization, and to design training sets that produce the best performance. This experiment, based on a separately and independently constructed evaluation set, is designed to evaluate the performance of NB on a set of documents that have never been submitted to the classifier. For this purpose, we further carefully collected manually 10 documents from Aljazeera.net for each of the 5 predefined categories. For each category, we have selected documents that best represent the variability in the category. We refer to this collection of documents as the evaluation set. This set is presented to the classifier for categorization.

For testing on the evaluation set, trained NB classifiers are used. For each category, we use the NB classifier that has been trained using the training set that produced the best category classification accuracy in cross validation experiments. In our case, we have used the whole set as a training set (1,500) represented by 2,000 terms since the best cross validation accuracy was obtained in leave-one-out experiment with 2,000 terms. Table 7 summarizes NB's performance results when tested using the evaluation set. The results obtained have shown higher performance for the Sports and the Business categories with a classification accuracy that is higher than 70%. The performance of other categories ranges from 40% to 60%. The average accuracy over all categories is 62%.

The results obtained in the evaluation set experiment are very consistent with the performance obtained in cross validation experiments.

6 Conclusions

To sum up, this work has been carried out to automatically classify Arabic documents using the NB algorithm, with the use of a different data set, a different number of categories, and a different root extraction algorithm from those used in (Yahyaoui, 2001). In this work, the average accuracy over all categories is: 68.78% in cross validation and 62% in evaluation set experiments. The corresponding performances in (Yahyaoui, 2001) are 75.6% and 50%, respectively. Thus, the overall performance (including cross validation and evaluation set experiments) in this work is comparable to that in (Yahyaoui, 2001). This offers some indication that the performance of NB algorithm in classifying Arabic documents is not sensitive to the Arabic root extraction algorithm. Future work will be directed at experimenting with other root extraction algorithms. Further improvement of NB's performance may be effected by using unlabeled documents; e.g., EM has been used successfully for this purpose in (Nigam et al., 200), where EM has increased the classification accuracy by 30% for classifying English documents. Two (English) document categorization algorithms have been reported to produce best results: Support Vector Machines (SVM) and AdaBoost. If the similarity between NB's performance for English and Arabic is any indication, SVM and AdaBoost should be the next candidates for application to Arabic Document categorization.

References

R. Al-Shalabi, and M. Evens, "A computational morphology system for Arabic," *In Workshop on Computational Approaches to Semitic Languages*, COLING-ACL98, 1998.

B. Cestink, "Estimating probabilities: A crucial task in machine learning," *Proceedings of the Ninth European Conference on Artificial Intelligence*, pp. 147--149, London, 1990.

K. Crammer and Y. Singer, "A Family of Additive Online Algorithms for Category Ranking," *JMLR,* v. 3, pp. 1025-1058, Feb. 2003.

R. H. Creecy, B. M. Masand, S. J. Smith, and D. L. Waltz, "Trading mips and memory for knowledge engineering," *Communication of the ACM*, Vol. 35, No. 8, pp. 48--64, August 1992.

M. El Kourdi, T. Rachidi, and A. Bensaid, "*A concatenative approach to Arabic word root extraction*," in progress, 2004.

Y.C. Fang, S. Parthasarathy and F. Schwartz, "Using clustering to boost text classification," *ICDM Workshop on Text Mining (TextDM'01)*, 2001.

Y. Houmame, *Towards an Arabic Information Retrieval System*, MS thesis, AlAkhawayn University, Morocco, 1999.

T. Joachims, *Learning to classify text using SVM*, Kluwer Academic Publishers, 2002.

K. Lang, "Newsweeder: Learning to filter netnews," *Proceedings of the Twelfth International Conference on Machine Learning,* 1995.

D. Lewis, M. Ringnette, "Comparison of two learning algorithms for text categorization," *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, 1994.

D. Lewis, Yiming Yang, Tony G. Rose, Fan Li, "A New Benchmark Collection for Text Categorization Research," JMLR, v. 5, pp. 361-397, Apr. 2004.

T. Mitchell. Machine learning. McGraw Hill, 1997.

K. Nigam, A. K. McCallum, S. Thrun, and T.Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, pp. 103--134, 2000.

T. Rachidi, O. Iraqi, M. Bouzoubaa, A. Ben Al Khattab, M. El Kourdi, A. Zahi, and A. Bensaid, "Barq: distributed multilingual Internet search engine with focus on Arabic language," *Proceedings of IEEE Conf. on Sys., Man and Cyber., Washington DC, October 5-8, pp. , 2003.*

M. Rogati and Y. Yang. "High-performing feature selection for text classification," ACM CIKM 2002.

Sakhr software company's website: www.sakhrsoft.com, 2004.

G. Salton and C. S. Yang, "On the specification of term values in automatic indexing", *Journal of Documentation*, Vol. 29, No. 4, pp. 351--372, 1973.

F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, v.34 n.1, p.1-47, March 2002.

K. Tzeras and S. Hartman, "Automatic indexing based on Bayesian inference networks," *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pp. 22--34, 1993.

(Wiene and Pedersen, 1995) E. Wiener, J. O. Pedersen, and A. S. Zeigend, "A neural network approach to topic spotting," *Proceedings of the Fourth Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, 1995.

M. Yahyaoui, "*Toward an Arabic web page classifier*," Master project. AUI. 2001.

Y. Yang, "An evaluation of statistical approaches to text categorization," *Journal of Information Retrieval*, Vol. 1, Number 1-2, pp. 69--90, 1999.

Y. Yang and X. Liu, "A re-examination of text categorization methods," *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR'99), pp 42--49, 1999.

R. B. Yates, and B. R. Neto, *Modern information retrieval*. Addison-Wesley ISBN 0-201-39829-X, 1999.

Yang, Y., Pedersen J.P. A Comparative Study on Feature Selection in Text Categorization *Proceedings of the 14th International Conference on Machine Learning*, pp. 412-420, 1997.

A Transcription Scheme for Languages Employing the Arabic Script Motivated by Speech Processing Application

Shadi GANJAVI
*Department of Linguistics
University of Southern California
ganajvi@usc.edu

Panayiotis G. GEORGIOU,
Shrikanth NARAYANAN*
Department of Electrical Engineering
Speech Analysis & Interpretation
Laboratory (sail.usc.edu)
[georgiou, shri]@sipi.usc.edu

Abstract

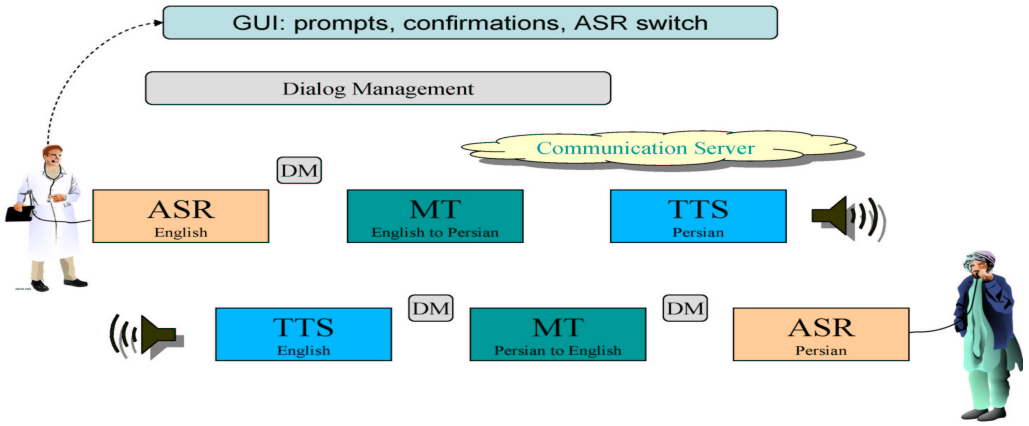
This paper offers a transcription system for Persian, the target language in the Transonics project, a speech-to-speech translation system developed as a part of the DARPA Babylon program (The DARPA Babylon Program; Narayanan, 2003). In this paper, we discuss transcription systems needed for automated spoken language processing applications in Persian that uses the Arabic script for writing. This system can easily be modified for Arabic, Dari, Urdu and any other language that uses the Arabic script. The proposed system has two components. One is a phonemic based transcription of sounds for acoustic modelling in Automatic Speech Recognizers and for Text to Speech synthesizer, using ASCII based symbols, rather than International Phonetic Alphabet symbols. The other is a hybrid system that provides a minimally-ambiguous lexical representation that explicitly includes vocalic information; such a representation is needed for language modelling, text to speech synthesis and machine translation.

1 Introduction

Speech-to-speech (S2S) translation systems present many challenges, not only due to the complex nature of the individual technologies involved, but also due to the intricate interaction that these technologies have to achieve. A great challenge for the specific S2S translation system involving Persian and English would arise from not only the linguistics differences between the two languages but also from the limited amount of data available for Persian. The other major hurdle in achieving a S2S system involving these languages is the Persian writing system, which is based on the Arabic script, and hence lacks the explicit inclusion of vowel sounds, resulting in a very large amount of one-to-many mappings from transcription to acoustic and semantic representations.

In order to achieve our goal, the system that was designed comprised of the following components:

Fig 1. Block diagram of the system. Note that the communication server allows interaction between all subsystems and the broadcast of messages. Our vision is that only the doctor will have access to the GUI and



the patient will only be given a phone handset.

(1) a visual and control Graphical User Interface (GUI); (2) an Automatic Speech Recognition (ASR) subsystem, which works both using Fixed State Grammars (FSG) and Language Models (LM), producing n-best lists/lattices along with the decoding confidence scores; (3) a Dialog Manager (DM), which receives the output of the speech recognition and machine translation units and subsequently "re-scores" the data according to the history of the conversation; (4) a Machine Translation (MT) unit, which works in two modes: Classifier based MT and a fully Stochastic MT; and finally (5) a unit selection based Text To Speech synthesizer (TTS), which provides the spoken output. A functional block diagram is shown in Figure 1.

1.1 The Language Under Investigation: Persian

Persian is an Indo-European language with a writing system based on the Arabic script. Languages that use this script have posed a problem for automated language processing such as speech recognition and translation systems. For instance, the CSLU Labeling Guide (Lander, http://cslu.cse.ogi.edu/corpora/corpPublications.html) offers orthographic and phonetic transcription systems for a wide variety of languages, from German to Spanish with a Latin-based writing system to languages like Mandarin and Cantonese, which use Chinese characters for writing. However, there seems to be no standard transcription system for languages like Arabic, Persian, Dari, Urdu and many others, which use the Arabic script (ibid; Kaye, 1876; Kachru, 1987, among others).

Because Persian and Arabic are different, Persian has modified the writing system and augmented it in order to accommodate the differences. For instance, four letters were added to the original system in order to capture the sounds available in Persian that Arabic does not have. Also, there are a number of *homophonic* letters in the Persian writing system, i.e., the same sound corresponding to different orthographic representations. This problem is unique to Persian, since in Arabic different orthographic representations represent different sounds. The other problem that is common in all languages using the Arabic script is the existance of a large number of *homographic* words, i.e., orthographic representations that have a similar form but different pronunciation. This problem arises due to limited vowel presentation in this writing system.

Examples of the homophones and homographs are represented in Table 1. The words "six" and "lung" are examples of homographs, where the identical (transliterated Arabic) orthographic representations (Column 3) correspond to different pronunciations [SeS] and [SoS] respectively (Column 4). The words "hundred" and "dam" are examples of homophones, where the two words have similar pronunciation [sad] (Column 4), despite their different spellings (Column 3).

	Persian	**USCPers**	**USCPron**	**USCPers+**
'six'	شش	SS	SeS	SeS
'lung'	شش	SS	SoS	SoS
'100'	صد	$d	sad	$ad
'dam'	سد	sd	sad	sad

Table 1 Examples of the transcription methods and their limitation. Purely orthographic transcription schemes (such as USCPers) fail to distinctly represent homographs while purely phonetic ones (such as USCPron) fail to distinctly represent the homophones.

The former is the sample of the cases in which there is a many-to-one mapping between orthography and pronunciation, a direct result of the basic characteristic of the Arabic script, viz., little to no representation of the vowels.

As is evident by the data presented in this table, there are two major sources of problems for any speech-to-speech machine translation. In other words, to employ a system with a direct 1-1 mapping between Arabic orthography and a Latin based transcription system (what we refer to as USCPers in our paper) would be highly ambiguous and insufficient to capture distinct words as required by our speech-to-speech translation system, thus resulting in ambiguity at the text-to-speech output level, and internal confusion in the language modelling and machine translation units. The latter, on the other hand, is a representative of the cases in which the same sequence of sounds would correspond to more than one orthographic representation. Therefore, using a pure phonetic transcription, e.g., USCPron, would be acceptable for the *Automatic Speech Recognizer* (ASR), but not for the *Dialog Manager* (DM) or the *Machine Translator* (MT). The goal of this paper is twofold (i) to provide an ASCII based phonemic transcription system similar to the one used in the International Phonetic Alphabet (IPA), in line of Worldbet (Hieronymus, http://cslu.cse.ogi.edu/corpora/corpPublications.html) and (ii) to argue for an ASCII based hybrid

transcription scheme, which provides an easy way to transcribe data in languages that use the Arabic script.

We will proceed in Section 2 to provide the USCPron ASCII based phonemic transcription system that is similar to the one used by the International Phonetic Alphabet (IPA), in line of Worldbet (ibid). In Section 3, we will present the USCPers orthographic scheme, which has a one-to-one mapping to the Arabic script. In Section 4 we will present and analyze USCPers+, a hybrid system that keeps the orthographic information, while providing the vowels. Section 5 discusses some further issues regarding the lack of data.

2 Phonetic Labels (USCPron)

One of the requirements of an ASR system is a phonetic transcription scheme to represent the pronunciation patterns for the acoustic models. Persian has a total of 29 sounds in its inventory, six vowels (Section 2.1) and 23 consonants (Section 2.2). The system that we created to capture these sounds is a modified version of the International Phonetic Alphabet (IPA), called USCPron(unciation). In USCPron, just like the IPA, there is a one-to-one correspondence between the sounds and the symbols representing them. However, this system, unlike IPA does not require special fonts and makes use of ASCII characters. The advantage that our system has over other systems that use two characters to represent a single sound is that following IPA, our system avoids all ambiguities.

2.1 Vowels

Persian has a six-vowel system, high to low and front and back. These vowels are: [i, e, a, u, o, A], as are exemplified by the italicized vowels in the following English examples: 'b*ea*t', 'b*e*t', 'b*a*t', 'p*u*ll', 'p*o*ll' and 'p*o*t'. The high and mid vowels are represented by the IPA symbols. The low front vowel is represented as [a], while the low back vowel is represented as [A]. There are no diphthongs in Persian, nor is there a tense/lax distinction among the vowels (Windfuhr, Gernot L.1987).

	Front	Back
High	i	u
Mid	e	o
Low	a	A

Table 2: Vowels

2.2 Consonants

In addition to the six vowels, there are 23 distinct consonantal sounds in Persian. Voicing is phonemic in Persian, giving rise to a quite symmetric system. These consonants are represented in Table 3 based on the place (bilabial (BL), lab-dental (LD), dental (DE), alveopalatal (AP), velar (VL), uvular (UV) and glottal (GT)) and manner of articulation (stops (ST), fricatives (FR), affricates (AF), liquids (LQ), nasals (NS) and glides (GL)) and their voicing ([-v(oice)] and [+v(oice)].

	BL	LD	DE	AP	VL	UV	GT
ST [-v]	p		t		k		?
[+v]	b		d		g	q	
FR [-v]		f	s	S	x		h
[+v]		v	z	Z			
AF [-v]				C			
[+v]				J			
LQ			l, r				
NS	m		n				
GL				y			

Table 3: Consonants

Many of these sounds are similar to English sounds. For instance, the stops, [p, b, t, d, k, g] are similar to the italicized letters in the following English words: '*p*otato', '*b*all', '*t*ree', '*d*oll', '*k*ey' and 'do*g*' respectively. The glottal stop [?] can be found in some pronunciations of 'bu*tt*on', and the sound in between the two syllables of 'uh oh'. The uvular stop [q] does not have a correspondent in English. Nor does the velar fricative [x]. But the rest of the fricatives [f, v, s, z, S, Z, h] have a corresponding sound in English, as demonstrated by the following examples '*f*ine', '*v*alue', '*s*and', '*z*ero', '*sh*ore', 'plea*su*re' and '*h*ello'. The affricates [C] and [J] are like their English counterparts in the following examples: '*ch*ur*ch*' and '*j*udge'. The same is true of the nasals [m, n] as in '*m*ake' and '*n*o'; liquids [r, l], as in '*r*ain' and '*l*ong' and the glide [y], as in '*y*esterday'. (The only distinction between Persian and English is that in Persian [t, d, s, z, l, r, n] are dental sounds, while in English they are alveolar.) As is evident, whenever possible, the symbols used are those of the International Phonetic Alphabet (IPA).

However, as mentioned before because IPA requires special fonts, which are not readily available for a few of the sounds, we have used an ASCII symbol that resembled the relevant IPA

symbol. The only difference between our symbols and the ones used by IPA are in voiceless and voiced alveopalatal fricatives [S] and [Z], the voiceless and voiced affricates [C] and [J], and the palatal glide [y]. In the case of the latter, we did not want to use the lower case 'j', in order to decrease confusion.

3 Orthographic Labels (USCPers)

We proceed in this section to present an alternative orthographic system for Persian, as a first step in the creation of the USCPers+ system that will be presented later. The Persian writing system is a consonantal system with 32 letters in its alphabet (Windfuhr, 1987). All but four of these letters are direct borrowing from the Arabic writing system. It is important to note that this borrowing was not a total borrowing, i.e., many letters were borrowed without their corresponding sound. This has resulted in having many letters with the same sound (homophones). However, before discussing these cases, let us consider the cases in which there is no homophony, i.e., the cases in which a single letter of the alphabet is represented by a single sound.

In order to assign a symbol to each letter of the alphabet, the corresponding letter representing the sound of that letter was chosen. So, for instance for the letter 'پ', which is represented as [p] in USCPron, the letter 'p' was used in USCPers(ian).

These letters are:

ST	FR	AF	LQ	NS
پ p	ف f	چ C	ر r	م m
ب b	ش S	ج J	ل l	ن n
د d	ژ Z			
ک k	خ x			
گ g				
ع ?				

Table 4: USCPers(ian) Symbols: Non-Homophonic Consonants

As mentioned above, this partial borrowing of the Arabic writing system has given rise to many homophonic letters. In fact, thirteen letters of the alphabet are represented by only five sounds. These sounds and the corresponding letters are presented below:

- [t] for 'ت' and 'ط';
- [q] for 'ق' and 'غ';
- [h] for 'ه' and 'ح';
- [s] for 'س', 'ص' and 'ث' and
- [z] for 'ز', 'ذ', 'ض', and 'ظ'.

In these cases, several strategies were used. If there were two letters with the same sound, the lower case and the upper case letters were used, as in table 5. In all these cases, the lower case letter is assigned to the most widely used letter and the upper case, for the other.

[t]	ت	t	ط	T
[q]	ق	q	غ	Q
[h]	ه	h	ح	H

Table 5 USCPers(ian) Symbols: Homophonic Consonants 1

In the case of the letters represented as [s] and [z] in USCPron, because the corresponding upper case letters were already assigned, other symbols were chosen. For the letters sounding [s], 's', '$' and '&' and for the letters sounding [z], 'z', '2', '7' and '#'.

[s]	س	s	ص	$	ث	&			
[z]	ز	z	ض	2	ظ	7	ذ	#	

Table 6 USCPers(ian) Symbols: Homophonic Consonants 2

These letters are not the only ambiguous letters in Persian. The letters 'ی' and 'و' can be used as a consonant as well as a vowel, [y] and [i] in the case of the former and [v], [o] and [u] in the case of the latter. However, in USCPers, the symbols 'y' and 'v' were assigned to them, leaving the pronunciation differences for USCPron to capture. For instance, the word for 'you' is written as 't**v**' in USCPers, but pronounced as [t**o**], and the word 'but' is written as '**v**ly' and pronounced as [**v**ali].

As is the characteristics of languages employing the Arabic script, for the most part the vowels are not represented and Persian is no exception. The only letter in the alphabet that represents a vowel is the letter 'alef'. This letter has different appearances depending on where it appears in a word. In the word initial position, it appears as 'آ', elsewhere it is represented as 'ا'. Because the dominant sound that this letter represents is the sound [A], the letter 'A' was assigned to represent 'ا', which has a wider distribution; 'V' was assigned for the more restricted version 'آ'. In Persian, like in Arabic, diacritics mark the vowels, although they are not used in writing, unless to avoid ambiguities. Therefore, in our system, we ignored the diacritics.

Borrowed Letters	USCPers Symbol	USC-Pron
أ	@	an
أ	*	a
ئ	Y	e
ء	^	no sound
ؤ	W	o

Table 7 Non-Persian Letters

Finally in creating the one-to-one mapping between the Persian alphabet and USCPers, we need to deal with the issue of "pure Arabic" letters that appear in a handful of words. We see the same situation in the borrowed words in English, for instance the italicized letters in ca*ñ*on or na*ï*ve, are not among the letters of the English alphabet, but they appear in some words used in English. In order to ensure a one-to-one representation between the orthography and USCPers, these letters were each assigned a symbol, as presented on Table7.

USCPers, therefore, provides us with a way to capture each letter of the alphabet with one and only one ASCII symbol, creating a comparable system to USCPron for the orthography.

4 USCPers/USCPron: Two Way Ambiguity

As was noted in the previous section, vowels are not usually represented in orthography and there are many homophonic letters. These two properties can give rise to two sources of ambiguity in Persian which can pose a problem for speech-to-speech machine translation: (i) in which two distinct words have the same pronunciation (homophones), like 'pair' and 'pear' in English and the Persian words like 'sd' and '$d', which are both pronounced as [sad] and (ii) in which one orthographic representation can have more than one pronunciation (homographs) similar to the distinction between the two English words convict (n) and convict (v), which are both spelled c-o-n-v-i-c-t, but different stress assignments create different pronunciations. It is important to note that English has a handful of such homographic pairs, while in Persian homographs are very common, contributing to much ambiguity. In this section, we will discuss the transcription system we have adopted in order to eliminate these ambiguities.

4.1 Homophones

The examples in Table 8 illustrate the case in (i) (the letters with the same sounds are underlined). As is evident by the last column in Table 8, in each case, the two words have similar pronunciation, but different spellings.

Gloss	USCPers	USCPron
'hundred'	$d	[sad]
'dam'	sd	[sad]
'life'	HyAt	[hayAt]
'backyard'	HyAT	[hayAt]
'Eve'	HvA	[havA]
'air'	hvA	[havA]

Table 8: Same Pronunciation, Different Spellings

The word for 'life' ends in 't', while the word for 'backyard' ends in 'T'. In the other examples, because there is no difference in the pronunciation of 'h'/'H' and 's'/'$', we get ambiguity between 'Eve'/'air' and 'hundred'/'dam'. Therefore, this type of ambiguity appears only in speech.

4.2 Homographs

The second case of ambiguity is illustrated by the examples in the following table:

Gloss	USCPers	USCPron
'lung'	SS	[SoS]
'six'	SS	[SeS]
'thick'	klft	[koloft]
'maid'	klft	[kolfat]
'Cut!'	bbr	[bebor]
'tiger'	bbr	[babr]

Table 9: Same Spelling, Different Pronunciations

Here, we see that in the middle column two words that have the same orthographic representation correspond to different pronunciations (Column 3), marking different meanings, as is indicated by the gloss. This type of ambiguity arises only in writing and not speech.

4.3 Solution: USCPers+

Because of the ambiguity presented by the lack of vowels the data transcribed in USCPers cannot be used either by MT or for language modeling in ASRs, without significant loss of information. In order to circumvent this problem, we adopted a

modified version of USCPers. In this new version, we have added the missing vowels, which would help to disambiguate. (Because this new version is USCPers + vowels, it is called USCPers+.) In other words, USCPers+ provides both the orthographic information as well as some phonological information, giving rise to unique words. Let us reconsider the examples we saw above using this new transcription system. A modified version of Table 8 is presented in Table 10.

Gloss	USCPers	USCPers+	USCPron
'hundred'	$d	$ad	[sad]
'dam'	sd	sad	[sad]
'life'	HyAt	HayAt	[hayAt]
'backyard'	HyAT	HayAT	[hayAt]
'Eve'	HvA	HavA	[havA]
'air'	hvA	havA	[havA]

Table 10: USPers+ Disambiguates Cases with Same Pronunciation & Different Spellings

Table 11 is the modified version of Table 9:

Gloss	USCPers	USCPers+	USCPron
'lung'	SS	SoS	[SoS]
'six'	SS	SeS	[SeS]
'thick'	klft	koloft	[koloft]
'maid'	klft	kolfat	[kolfat]
'Cut!'	bbr	bebor	[bebor]
'tiger'	bbr	babr	[babr]

Table 11: USCPers+ Disambiguates Cases with Same Spelling & Different Pronunciations

Data in Column 4 and Column 2 of Tables 10 and 11, respectively, show that USCPron and USCPers can give rise to ambiguity, while no ambiguity exists in USCPers+, Column 3.

The following sentence also illustrates this point, where the words 'thick' and 'maid' from Table 11 are used. Assume that ASR receives the audio input in (1) represented in USCPron:

(1) USCPron: [in koloft ast]
Gloss: thisthick is
Translation: 'This is thick'

If ASR outputs USCPers, as in (2),

(2) USCPers: Ayn klft Ast

the MT output in the English language can choose either:

(3) a. This is thick
b. This is a maid

as a possible translation. However, using USCPers+ instead of USCPers would avoid this ambiguity:

(4) USCPers+: Ayn koloft Ast (cf. (2))

As evident, there is a significant benefit by using USCPers+.

The discussion of the conventions that have been adopted in the use of USCPers+ and USCPron, e.g., not including punctuations or spelling out numbers, is beyond the scope of this paper. However, it is important to note that by adopting a reasonable number of conventions in our transcription of USCPers+ and USCPron, we have been able to provide a complete transcription convention for acoustic models and language models for the ASRs, TTSs and MTs for our English to Persian translation system.

5 Further Issue: Dealing with the Lack of Data

Despite the significant advantages of employing the USCPers+ transcription scheme, a drawback is the lack of data in this format. To address this shortcoming, semi-automated techniques of data conversion have been developed that take into consideration the statistical structure of the language. Fig. 2 depicts a network that can be inferred from a relatively small amount of humanly transliterated data. By employing statistical decoding techniques through such a model, the most likely USCPers+ sequence can be generated using minimal human intervention.

Consider for example the sentence 'SS mn drd myknd' and the network structure shown above. It is likely that the combination 'man dard' and 'dard mykonad' have been seen in the manually generated data, and thus the decoder is likely to chose the path 'man dard mykonad' as the correct transliteration.

Manual decision can be made in the cases that the system reaches a statistical ambiguity (usually in cases such as 'Ayn klft Ast') or that insufficient training data exist for the specific region of decoding.

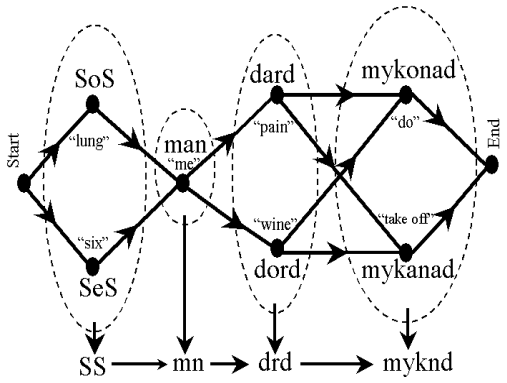


Fig 2. The possible transitions between words are probabilistically denoted in a language model, which can be employed for decoding of the most likely path, given several possibilities. Shown above are the possibilities for the decoding of the utterance "SS mn drd myknd".

The first ambiguity is rare, and usually involves short segments of text. Thus as the models improve, and we move to higher orders of decoding, the statistical ambiguity becomes less significant. Similarly, the unknown words keep decreasing as new converted data feeds back into the training corpus.

In our experiments, as the amount of training data grew from about 16k to 22k words, the precision in transliteration increased from 98.85% to 99.2%, while at the same time the amount of manual intervention was reduced from 39.6% to 22%. It should be noted that by changing the decision thresholds the intervention can fall significantly lower, to 9.4% with a training corpus of 22k words, but this has the effect of a lower precision in the order of 98.8%.

An indepth discussion of the techniques employed for the transliteration process is presented in Georgiou, et.al (2004).

6 Conclusion

This paper argues that the best way to represent data at phonological/lexical level for language modeling and MT in languages that employ the Arabic script, is by using a hybrid system, which combines information provided by orthography and includes the vowels that are not represented in orthography. The schemes proposed can significantly aid in speech-to-speech applications in a multitude of different ways: (1) the internal pronunciations of the ASR and the TTS components can employ the USCPron scheme, (2) the internal transcription of the Persian language—for purposes of language modeling and statistical machine translation among others—can employ the USCPers+ scheme and (3) in the case of a stand-alone TTS, in which case the input is pure Persian text, automated transliteration to the USCPers+ scheme, and hence to the pronunciation, can be generated with statistical language augmentation techniques, which are based on prior model training, as we describe further in Georgiou, 2004.

This would ensure a uniqueness that otherwise is not available. It has also been suggested in this paper that a modification of IPA, which would allow the use of ASCII characters, is a more convenient way to capture data for acoustic modeling and TTS. Persian data resources developed under the DARPA Babylon program have adopted the conventions described in this paper.

7 Acknowledgements

This work was supported by the DARPA Babylon program, contract N66001-02-C-6023. We would like to thank the following individuals for their comments and suggestion: Naveen Srinivasamurthy and HS, MK and SS for working with the first versions of this system and making insightful suggestions.

8 References

The DARPA Babylon program," http://darpa-babylon.mitre.org.

P. Georgiou, H. Shiranimehr and S. Narayanan (2004). Context Dependent Statistical Augmentation of Persian Transcripts for use in Speech to Speech Translation Applications. INTERSPEECH 2004-International Conference on Spoken Language Processing.

J.L. Hieronymus, ASCII Phonetic Symbols for the World's Languages: Worldbet, AT&T Bell Labs, http://cslu.cse.ogi.edu/corpora/corpPublications.html

Y. Kachru. 1987. "Hindi-Urdu," The World's Major Languages, ed. Bernard Comrie, Oxford University Press.

A.S. Kaye. 1987. "Arabic," The World's Major Languages, ed. Bernard Comrie, Oxford University Press.

T. Lander, The CSLU Labeling Guide, OGI, http://cslu.cse.ogi.edu/corpora/corpPublications.html

S. Naraynan, et. al. 2003. Transonics: A speech to speech system for English-Persian interactions.

G.L. Windfuhr. (1987). "Persian," The World's Major Languages, ed. Bernard Comrie, Oxford University Press.

Automatic diacritization of Arabic for Acoustic Modeling in Speech Recognition

Dimitra Vergyri
Speech Technology and Research Lab.,
SRI International,
Menlo Park, CA 94025, USA
dverg@speech.sri.com

Katrin Kirchhoff
Department of Electrical Engineering,
University of Washington,
Seattle, WA 98195, USA
katrin@ee.washington.edu

Abstract

Automatic recognition of Arabic dialectal speech is a challenging task because Arabic dialects are essentially spoken varieties. Only few dialectal resources are available to date; moreover, most available acoustic data collections are transcribed without diacritics. Such a transcription omits essential pronunciation information about a word, such as short vowels. In this paper we investigate various procedures that enable us to use such training data by automatically inserting the missing diacritics into the transcription. These procedures use acoustic information in combination with different levels of morphological and contextual constraints. We evaluate their performance against manually diacritized transcriptions. In addition, we demonstrate the effect of their accuracy on the recognition performance of acoustic models trained on automatically diacritized training data.

1 Introduction

Large-vocabulary automatic speech recognition (ASR) for conversational Arabic poses several challenges for the speech research community. The most difficult problems in developing highly accurate speech recognition systems for Arabic are the predominance of non-diacritized text material, the enormous dialectal variety, and the morphological complexity.

Most available acoustic training material for Arabic ASR is transcribed in the Arabic script form, which does not include short vowels and other diacritics that reflect differences in pronunciation, such as the shadda, tanween, etc. In particular, almost all additional text data that can easily be obtained (e.g. broadcast news corpora) is represented in standard script form. To our knowledge, the only available corpus that does include detailed phonetic information is the CallHome (CH) Egyptian Colloquial Arabic (ECA) corpus distributed by the Linguistic Data Consortium (LDC). This corpus has been transcribed in both the script form and a so-called romanized form, which is an ASCII representation that includes short vowels and other diacritic information and thus has complete pronunciation information. It is quite challenging to create such a transcription: native speakers of Arabic are not used to writing their language in a "romanized" form, or even in fully diacritized script form. Consequently, this task is considered almost as difficult as phonetic transcription. Transcribing a sufficiently large amount of training data in this way is therefore labor-intensive and costly since it involves (re)-training native speakers for this purpose.

The constraint of having mostly non-diacritized texts as recognizer training material leads to problems for both acoustic and language modeling. First, it is difficult to train accurate acoustic models for short vowels if their identity and location in the signal is not known. Second, the absence of diacritics leads to a larger set of linguistic contexts for a given word form; language models trained on non-diacritized material may therefore be less predictive than those trained on diacritized texts. Both of these factors may lead to a loss in recognition accuracy. Previous work (Kirchhoff et al., 2002; Lamel, 2003) has shown that ignoring available vowel information does indeed lead to a significant increase in both language model perplexity and word error rate. Therefore, we are interested in automatically deriving a diacritized transcription from the Arabic script representation when a manual diacritization is not available. Some software companies (Sakhr, Apptek, RDI) have developed commercial products for the automatic diacritization of Arabic. However, these products use only text-based information, such as the syntactic context and possible morphological analyses of words, to predict diacritics. In the context of diacritization for speech recognition, by contrast, acoustic data is available that can be used as an additional knowledge source. Moreover, commer-

cial products concentrate exclusively on Modern Standard Arabic (MSA), whereas a common objective of Arabic ASR is conversational speech recognition, which is usually dialectal. For this reason, a more flexible set of tools is required in order to diacritize dialectal Arabic prior to speech recognizer training.

In this work we investigate the relative benefits of a variety of knowledge sources (acoustic, morphological, and contextual) to automatically diacritize MSA transcriptions. We evaluate the different approaches in two different ways: (a) by comparing the automatic output against a manual reference diacritization and computing the diacritization error rate, and (b) by using automatically diacritized training data in a cross-dialectal speech recognition application.

The remainder of this paper is structured as follows: Section 2 gives a detailed description of the motivation as well as prior work. Section 3 describes the corpora used for the experiments reported in this paper. The automatic diacritization procedures and results are explained in Section 4. The speech recognition experiments and results are reported in Section 5. Section 6 presents our conclusions.

2 Motivation and Prior Work

We first describe the Arabic writing system and its inherent problems for speech recognizer training, and then discuss previous attempts at automatic diacritization.

2.1 The Arabic Writing System

The Arabic alphabet consists of twenty-eight letters, twenty-five of which represent consonants and three of which represent the long vowels (/i:/,/a:/,/u:/). A distinguishing feature of Arabic-script based writing systems is that short vowels are not represented by the letters of the alphabet. Instead, they are marked by so-called *diacritics*, short strokes placed either above or below the preceding consonant. Several other pronunciation phenomena are marked by diacritics, such as consonant doubling (phonemic in Arabic), which is indicated by the "shadda" sign, and the "tanween", i.e. word-final adverbial markers that add /n/ to the pronunciation of the word. These diacritics are listed in Table 1. Arabic texts are almost never fully diacritized; normally, diacritics are used sparingly and only to prevent misunderstandings. Exceptions are important religious and/or political texts or beginners' texts for students of Arabic. The lack of diacritics may lead to considerable lexical ambiguity that must be resolved by contextual information, which in turn presupposes knowledge of the language. It was observed in (Debili et al., 2002) that a non-diacritized dictionary word form has 2.9 possible diacritized forms on average and that an Arabic text containing 23,000 word forms showed an average ratio of 1:11.6. The form كتب, for instance, has 21 possible diacritizations. The correspondence between graphemes and phonemes is relatively transparent compared to other languages like English or French: apart from certain special graphemes (e.g. laam alif), the relationship is one to one. Finally, it is worth noting that the writing system described above is that of MSA. Arabic dialects are primarily oral varieties in that they do not have generally agreed-upon writing standards. Whenever there is the need to write down dialectal speech, speakers will try to approximate the standard system as far as possible and use a phonetic spelling for non-MSA or foreign words.

MSA	Symbol Name	Meaning
◌َ	fatHa	/a/
◌ِ	kasra	/i/
◌ُ	Damma	/u/
◌ّ	shadda	consonant doubling
دْرس	sukuun	vowel absence
◌ً	tanween al-fatHa	/an/
◌ٍ	tanween al-kasr	/in/
◌ٌ	tanween aD-Damm	/un/

Table 1: Arabic diacritics

The lack of diacritics in standard Arabic texts makes it difficult to use non-diacritized text for training since the location and identity of short vowels and other phonetic segments are unknown. One possible approach is to use acoustic models for long vowels and consonants only, where the acoustic signal portions corresponding to unwritten segments are implicitly incorporated into the acoustic models for consonants (Billa et al, 2002). However, this leads to less discriminative acoustic and language models. Previous work (Kirchhoff et al., 2002; Lamel, 2003) has compared the word error rates of two CH ECA recognizers: one trained on script transcriptions and another trained on romanized transcriptions. It was shown that the loss in information due to training on script forms

results in significantly worse performance: a relative increase in word error rate of almost 10% was observed.

It seems clear that diacritized data should be used for training Arabic ASR systems whenever possible. As explained above, however, it is very expensive to obtain manually transcribed data in a diacritized form. Therefore, the corpora that do include detailed transcriptions are fairly small and any dialectal data that might become available in the future will also very likely be of limited size. By contrast, it is much easier to collect publicly available data (e.g. broadcast news data) and to transcribe it in script form. In order to be able to take advantage of such resources, we need to restore short vowels and other missing diacritics in the transcription.

2.2 Prior Work

Various software companies have developed automatic diacritization products for Arabic. However, all of these are targeted towards MSA; to our knowledge, there are no products for dialectal Arabic. In a previous study (Kirchhoff et al., 2002) one of these products was tested on three different texts, two MSA texts and one ECA text. It was found that the diacritization error rate (percentage of missing and wrongly identified or inserted diacritics) on MSA ranged between 9% and 28%, depending on whether or not case vowel endings were counted. However, on the ECA text, the diacritization software obtained an error rate of 48%.

A fully automatic approach to diacritization was presented in (Gal, 2002), where an HMM-based bigram model was used for decoding diacritized sentences from non-diacritized sentences. The technique was applied to the Quran and achieved 14% word error (incorrectly diacritized words).

A first attempt at developing an automatic diacritizer for dialectal speech was reported in (Kirchhoff et al., 2002). The basic approach was to use a small set of parallel script and diacritized data (obtained from the ECA CallHome corpus) and to derive diacritization rules in an example-based way. This entirely knowledge-free approach achieved a 16.6% word error rate.

Other studies (El-Imam, 2003) have addressed problems of grapheme-to-phoneme conversion in Arabic, e.g. for the purpose of speech synthesis, but have assumed that a fully diacritized version of the text is already available.

Several knowledge sources are available for determining the most appropriate diacritization of a script form: analysis of the morphological structure of the word (including segmentation into stems, prefixes, roots and patterns), consideration of the syntactic context in which the word form appears, and, in the context of speech recognition, the acoustic data that accompanies the transcription. Specific dictionary information could in principle be added (such as information about proper names), but this knowledge source is ignored for the purpose of this study. All of the approaches described above make use of text-based information only and do not attempt to use acoustic information.

3 Data

For the present study we used two different corpora, the FBIS corpus of MSA speech and the LDC CallHome ECA corpus.

The FBIS corpus is a collection of radio newscasts from various radio stations in the Arabic speaking world (Cairo, Damascus, Baghdad) totaling approximately 40 hours of speech (roughly 240K words). The transcription of the FBIS corpus was done in Arabic script only and does not contain any diacritic information. There were a total of 54K different script forms, with an average of 2.5 different diacritizations per word.

The CallHome corpus, made available by LDC, consists of informal telephone conversations between native speakers (friends and family members) of Egyptian Arabic, mostly from the Cairene dialect region. The corpus consists of about 20 hours of training data (roughly 160K words) and 6 hours of test data. It is transcribed in two different ways: (a) using standard Arabic script, and (b) using a romanization scheme developed at LDC and distributed with the corpus. The romanized transcription contains short vowels and phonetic segments corresponding to other diacritics. It is not entirely equivalent to a diacritized Arabic script representation since it includes additional information. For instance, symbols particular to Egyptian Arabic were used (e.g. "g" for /g/, the ECA pronunciation of the MSA letter ج), whereas the script transcriptions contain MSA letters only. In general, the romanized transcription provides more information about actual pronunciation and is thus closer to a broad phonetic transcription.

4 Automatic Diacritization

We describe three techniques for the automatic diacritization of Arabic text data. The first combines acoustic, morphological and contextual information to predict the correct form, the second ignores contextual information, and the third is fully acoustics based. The latter technique uses no morphological or syntactic constraints, and allows for all possible items to be inserted at every possible position.

4.1 Combination of Acoustic, Morphological and Contextual Information

Most Arabic script forms can have a number of possible morphological interpretations, which often correspond to different diacritized forms. Our goal is to combine morphological knowledge with contextual information in order to identify possible diacritizations and assign probabilities to them. Our procedure is as follows:

1. Generate all possible diacritized variants for each word, along with their morphological analyses (tags).

2. Train an unsupervised tagger to assign probabilities to sequences of these morphological tags.

3. Use the trained tagger to assign probabilities to all possible diacritizations for a given utterance.

For the first step we used the Buckwalter stemmer, which is an Arabic morphological analysis tool available from the LDC. The stemmer produces all possible morphological analyses of a given Arabic script form; as a by-product it also outputs the concomitant diacritized word forms. An example of the output is shown in Figure 1. The next step was to train an unsupervised tagger on the output to obtain tag n-gram probabilities. The number of different morphological tags generated by applying the stemmer to the FBIS text was 763. In order to obtain a smaller tag set and to be able to estimate probabilities for tag sequences more robustly, this initial tag needed to be conflated to a smaller set. We adopted the set used in the LDC Arabic TreeBank project, which was also developed based on the Buckwalter morphological analysis scheme. The FBIS tags were mapped to TreeBank tags using longest common substring matching; this resulted in 392 tags. Further possible reductions of the tag set were investigated but it was found that too much clustering (e.g. of verb subclasses into a single verb class) could result in the loss of important information. For instance, the tense and voice features of verbs are strong predictors of the short vowel patterns and should therefore be preserved in the tagset.

```
LOOK-UP WORD: قبل (qbl)
SOLUTION 1: (qabola) qabola/PREP
  (GLOSS): + before +
SOLUTION 2: (qaboli) qaboli/PREP
  (GLOSS): + before +
SOLUTION 3: (qabolu) qabolu/ADV
  (GLOSS): + before/prior +
SOLUTION 4:(qibal) qibal/NOUN
  (GLOSS): + (on the) part of +
SOLUTION 5:(qabila)
  qabil/VERB_PERFECT+a/PVSUFF_SUBJ:3MS
  (GLOSS): + accept/receive/approve + he/it <verb>
SOLUTION 6: (qab˜ala)
  qab al/VERB_PERFECT+a/PVSUFF_SUBJ:3MS
  (GLOSS): + kiss + he/it <verb>
```

Figure 1: Sample output of Buckwalter stemmer showing the possible diacritizations and morphological analyses of the script form قبل (*qbl*). Lower-case *o* stands for sukuun (lack of vowel).

We adopted a standard statistical trigram tagging model:

$$P(t_0, \ldots, t_n | w_0, \ldots, w_n) = \prod_{i=0}^{n} P(w_i|t_i) P(t_i|t_{i-1}, t_{i-2}) \quad (1)$$

where t is a tag, w is a word, and n is the total number of words in the sentence. In this model, words (i.e. non-diacritized script forms) and morphological tags are treated as observed random variables during training. Training is done in an unsupervised way, i.e. the correct morphological tag assignment for each word is not known. Instead, all possible assignments are initially considered and the Expectation-Maximization (EM) training procedure iteratively trains the probability distributions in the above model (the probability of word given tag, $P(w_i|t_i)$, and the tag sequence probability, $P(t_i|t_{i-1}, t_{i-2})$) until convergence. During testing, only the word sequence is known and the best tag assignment is found by maximizing the probability in Equation 1. We used the graphical modeling toolkit GMTK (Bilmes and Zweig, 2002) to train the tagger. The trained tagger was then used to assign probabilities to all possible sequences of three successive mor-

phological tags and their associated diacritizations to all utterances in the FBIS corpus.

Using the resulting possible diacritizations for each utterance we constructed a word-pronunciation network with the probability scores assigned by the tagger acting as transition weights. These word networks were used as constraining recognition networks with the acoustic models trained on the CallHome corpus to find the most likely word sequence (a process called alignment). We performed this procedure with different weights on the tagger probabilities to see how much this information should be weighted compared to the acoustic scores. Results for weights 1 and 5 are reported below.

Since the Buckwalter stemmer does not produce case endings, the word forms obtained by adding case endings were included as variants in the pronunciation dictionary used by the aligner. Additional variants listed in the dictionary are the taa marbuta alternations /a/ and /at/. In some cases (approximately 1.5% of all words) the Buckwalter stemmer was not able to produce an analysis of the word form due to misspellings or novel words. These were mapped to a generic reject model.

4.2 Combination of Acoustic and Morphological Constraints

We were interested in separately evaluating the usefulness of the probabilistic contextual knowledge provided by the tagger, and the morphological knowledge contributed by the Buckwalter tool. To that end we used the word networks produced by the method described above but stripped the tagger probabilities, thus assigning uniform probability to all diacritized forms produced by the morphological analyzer. We used the same acoustic models to find the most likely alignment from the word networks.

4.3 Using only Acoustic Information

Similarly, we wanted to evaluate the importance of using morphological information versus only acoustic information to constrain the possible diacritizations. This is particularly interesting since, as new dialectal speech data become available, the acoustics may be the only information source. As explained above, existing morphological analysis tools such as the Buckwalter stemmer have been developed for MSA only.

For that purpose, we generated word networks that include all possible short vowels at each allowed position in the word and allowed all possible case endings. This means that after every consonant there are at least 5 different choices: no vowel (corresponding to the sukuun diacritic), /i/, /a/, /u/, or consonant doubling caused by a shadda sign. Combinations of shadda and a short vowel are also possible. Since we do not use acoustic models for doubled consonants in our speech recognizer, we ignore the variants involving shadda and allow only four possibilities after every word-medial consonant: the three short vowels or absence of a vowel. Finally, we include the three tanween endings in addition to these four possibilities in word-final position. As before, the taa marbuta variants are also included.

In this way, many more possible "pronunciations" are generated for a script form than could ever occur. The number of possible variants increases exponentially with the number of possible vowel slots in the word. For instance, for a longer word with 7 possible positions, more than 16K diacritized forms are possible, not even counting the possible word endings. As before, we use these large pronunciation networks to constrain our alignment with acoustic models trained on CallHome data and choose the most likely path as the output diacritization.

In principle it would also be possible to determine diacritization performance in the absence of acoustic information, using only morphological and contextual knowledge. This can be done by selecting the best path from the weighted word transition networks without rescoring the network with acoustic models. However, this would not lead to a valid comparison in our case because case endings are only represented in the pronunciation dictionary used by the acoustic aligner; they are not present in the weighted transition network and thus cannot be hypothesized unless the acoustic aligner is used.

4.4 Autodiacritization Error Rates

We measured the performance of all three methods by comparing the output against hand transcribed references on a 500 word subset of the FBIS corpus. These references were fully diacritized script transcriptions created by a native speaker of Arabic who was trained in orthographic transcription but not in phonetic transcription. The diacritization error rate was measured as the percentage of wrong diacritization decisions out of all possible decisions. In particular, an error occurs when:

- a vowel is inserted although the reference

transcription shows either sukuun or no diacritic mark at the corresponding position (insertion).

• no vowel is produced by the automatic procedure but the reference contains a vowel mark at the corresponding position (deletion).

• the short vowel inserted does not match the vowel at the corresponding position (substitution).

• in the case of tanween and taa marbuta endings, either the required consonants or vowels are missing or wrongly inserted. Thus, in the case of a taa marbuwta ending with a following case vowel /i/, for instance, both the /t/ and the /i/ need to be present. If either is missing, one error is assigned; if both are missing, two errors are assigned.

Results are listed in Table 2. The first column reports the error rate at the word level, i.e. the percentage of words that contained at least one diacritization mistake. The second column lists the diacritization error computed as explained above. The first three methods have a very similar performance with respect to diacritization error rate. The use of contextual information (the tagger probabilities) gives a slight advantage, although the difference is not statistically significant. Despite these small differences, the word error rate is the same for all three methods; this is because a word that contains at least one mistake is counted as a word error, regardless of the total number of mistakes in the word, which may vary from system to system. Using only acoustic information doubles the diacritization error rate and increases the word error rate to 50%. Errors result mostly from incorrect insertions of vowels (e.g. بَغدَاد → بَغُدَاد). Many of these insertions may stem from acoustic effects created by neighbouring consonants, that give a vowel-like quality to transitions between consonants. The main benefit of using morphological knowledge lies in the prevention of such spurious vowel insertions, since only those insertions are permitted which result in valid words. Even without the use of morphological information, the vast majority of the missing vowels are still identified correctly. Thus, this method might be of use when diacritizing a variety of Arabic for which morphological analysis tools are not available. Note that the results obtained here are not directly comparable to any of the works described in Section 2.2 since we used a data set with a much larger vocabulary size.

Information used	Word level	Character level
acoustic + morphological + contextual (tagger prob. weight=5)	27.3	13.24
acoustic + morphological + contextual (tagger prob. weight=1)	27.3	11.54
acoustic + morphological (tagger prob. weight=0)	27.3	11.94
acoustic only	50.0	23.08

Table 2: Automatic diacritization error rates (%).

5 ASR Experiments

Our overall goal is to use large amounts of MSA acoustic data to enrich training material for a speech recognizer for conversational Egyptian Arabic. The ECA recognizer was trained on the romanized transcription of the CallHome corpus described above and uses short vowel models. In order to be able to use the phonetically deficient MSA transcriptions, we first need to convert them to a diacritized form. In addition to measuring autodiacritization error rates, as above, we would like to evaluate the different diacritization procedures by investigating how acoustic models trained on the different outputs affect ASR performance.

One motivation for using cross-dialectal data is the assumption that infrequent triphones in the CallHome corpus might have more training samples in the larger MSA corpus. In (Kirchhoff and Vergyri, 2004) we demonstrated that it is possible to get a small improvement in this task by combining the scores of models trained strictly on CallHome (CH) with models trained on the combined FBIS+CH data, where the FBIS data was diacritized using the method described in Section 4.1. Here we compare that experiment with the experiments where the methods described in Sections 4.2 and 4.3 were used for diacritizing the FBIS corpus.

5.1 Baseline System

The baseline system was trained with only CallHome data (CH-only). For these experiments we used a single front-end (13 mel-frequency cepstral coefficients with first and second differences). Mean and variance as well as Vocal Tract Length (VTL) normalization were performed per conversation side for CH and per speaker cluster (obtained automatically) for FBIS. We trained non-crossword,

System	dev96	eval03
simple CH-only	56.1	42.7
RT-2003 CH-only	52.6	39.7

Table 3: CH-only baseline WER (%)

continuous-density, genonic hidden Markov models (HMMs) (Digalakis and Murveit, 1994), with 128 gaussians per genone and 250 genones. Recognition was done by SRI's DECIPHER^{TM} engine in a multipass approach: in the first pass, phone-loop adaptation with two Maximum Likelihood Linear Regression (MLLR) transforms was applied. A recognition lexicon with 18K words and a bigram language model were used to generate the first pass recognition hypothesis. In the second pass the acoustic models were adapted using constrained MLLR (with 6 transformations) based on the previous hypothesis. Bigram lattices were generated and then expanded using a trigram language model. Finally, N-best lists were generated using the adapted models and the trigram lattices. The final best hypothesis was obtained using N-best ROVER (**?**). This system is simpler than our best current recognition system (submitted for the NIST RT-2003 benchmark evaluations) (Stolcke et al., 2003) since we used a single front end (instead of a combination of systems based on different front ends) and did not include HLDA, cross-word triphones, MMIE training or a more complex language model. The lack of these features resulted in a higher error rate but our goal here was to explore exclusively the effect of the additional MSA training data using different diacritization approaches. Table 3 shows the word error rates of the system used for these experiments and the full system used for the NIST RT-03 evaluations. Our full system was about 2% absolute worse than the best system submitted for that task. This shows that even though the system is simpler we are not operating far from the state-of-the-art performance for this task.

5.2 ASR Systems Using FBIS Data

In order to investigate the effect of additional MSA training data, we trained a system similar to the baseline but used training data pooled from both corpora (CH+FBIS). After performing alignment of the FBIS data with the networks described in Section 4.1, 10% of the data was discarded since no alignments could be found. This could be due to segmentation problems or noise in the acoustic files. The remaining 90% were used for our experiments. In order to account for the fact that we had much more data, and also more dissimilar data, we increased the model size to 300 genones.

For training the CH+FBIS acoustic models, we first used the whole data set with weight 2 for CH utterances and 1 for FBIS utterances. Models were then MAP adapted on the CH-only data (Digalakis et al., 1995). Since training involves several EM iterations, we did not want to keep the diacritization fixed from the first pass, which used CH-only models. At every iteration, we obtain better acoustic models which can be used to re-align the data. Thus, for the first two approaches, where the size of the pronunciation networks is limited due to the use of morphological information, the EM forward-backward counts were collected using the whole diacritization network and the best diacritization path was allowed to change at every iteration. In the last case, where only acoustic information was used, the pronunciation networks were too large to be run efficiently. For this reason, we updated the diacritized references once during training by realigning the networks with the newer models after the first training iteration. As reported in (Kirchhoff and Vergyri, 2004) the CH+FBIS trained system by itself did not improve much over the baseline (we only found a small improvement on the eval03 testset) but it provided sufficiently different information, so that ROVER combination (Fiscus, 1997) with the baseline yielded an improvement. As we can see in Table 4, all diacritization procedures performed practically the same: there was no significant difference in the word error rates obtained after the combination with the CH-only baseline. This suggests that we may be able to obtain improvements with automatically diacritized data even when using inaccurate diacritization, produced without the use of morphological constraints.

6 Conclusions

In this study we have investigated different options for automatically diacritizing Arabic text for use in acoustic model training for ASR. A comparison of the different approaches showed that more linguistic information (morphology and syntactic context) in combination with the acoustics provides lower diacritization error rates. However, there is no significant difference among the word error rates of ASR sys-

	dev96		eval03	
System	alone	Rover with CH-only	alone	Rover with CH-only
CH-only	56.1		42.7	
CH+FBIS1(weight 1)	56.3	55.3	42.2	41.6
CH+FBIS1(weight 5)	56.1	55.2	42.2	41.8
CH+FBIS2	56.2	55.3	42.4	41.6
CH+FBIS3	56.6	55.7	42.1	41.6

Table 4: Word error rates (%) obtained after the final recognition pass and with ROVER combination with the baseline system. FBIS1, FBIS2 and FBIS3 correspond to the diacritization procedures described in Sections 4.1, 4.2 and 4.3 respectively. For the first approach we report results using the tagger probabilities with weights 1 and 5.

tems trained on data resulting from the different methods. This result suggests that it is possible to use automatically diacritized training data for acoustic modeling, even if the data has a comparatively high diacritization error rate (23% in our case). Note, however, that one reason for this may be that the acoustic models are finally adapted to the accurately transcribed CH-only data. In the future, we plan to apply knowledge-poor diacritization procedures to other dialects of Arabic, for which morphological analyzers do not exist.

7 Acknowledgments

This work was funded by DARPA under contract No. MDA972-02-C-0038. We are grateful to Kathleen Egan for making the FBIS corpus available to us, and to Andreas Stolcke and Jing Zheng for valuable advice on several aspects of this work.

References

J. Billa et al. 2002. Audio indexing of Broadcast News. In *Proceedings of ICASSP*.

J. Bilmes and G. Zweig. 2002. The Graphical Models Toolkit: An open source software system for speech and time-series processing. In *Proceedings of ICASSP*.

F. Debili, H. Achour, and E Souissi. 2002. De l'étiquetage grammatical à la voyellation automatique de l'arabe. Technical report, Correspondances de l'Institut de Recherche sur le Maghreb Contemporain.

V. Digalakis and H. Murveit. 1994. GENONES: Optimizing the degree of mixture tying in a large vocabulary hidden markov model based speech recognizer. In *Proceeding of ICASSP*, pages I–537–540.

V.V. Digalakis, D. Rtischev, and L. G. Neumeyer. 1995. Speaker adaptation using constrained estimation of gaussian mixtures. *IEEE Transactions SAP*, 3:357–366.

Yousif A. El-Imam. 2003. Phonetization of Arabic: rules and algorithms. *Computer, Speech and Language*, in press, preprint available online at www.sciencedirect.com.

J. G. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings IEEE Automatic Speech Recognition and Understanding Workshop*, pages 347–352, Santa Barbara, CA.

Ya'akov Gal. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 27–33, Philadelphia, July. Association for Computational Linguistics.

K. Kirchhoff and D. Vergyri. 2004. Cross-dialectal acoustic data sharing for Arabic speech recognition. In *Proceedings of ICASSP*.

K. Kirchhoff, J. Bilmes, J. Henderson, R. Schwartz, M. Noamany, P. Schone, G. Ji, S. Das, M. Egan, F. He, D. Vergyri, D. Liu, and N. Duta. 2002. Novel approaches to Arabic speech recognition - final report from the JHU summer workshop 2002. Technical report, Johns Hopkins University.

L. Lamel. 2003. Personal communication.

A. Stolcke, Y. Konig, and M. Weintraub. 1997. Explicit word error minimization in N-best list rescoring. In *Proceedings of Eurospeech*, volume 1, pages 163–166.

A. Stolcke et al. 2003. Speech-to-text research at sri-icsi-uw. Technical report, NIST RT-03 Spring Workshop. availble online http://www.nist.gov/speech/tests/rt/rt2003/spring/presentations/sri+-rt03-stt.pdf.

Letter-to-Sound Conversion for Urdu Text-to-Speech System

Sarmad HUSSAIN
Center for Research in Urdu Language Processing,
National University of Computer and Emerging Sciences
B Block, Faisal Town
Lahore, Pakistan
sarmad.hussain@nu.edu.pk

Abstract

Urdu is spoken by more than 100 million people across a score countries and is the national language of Pakistan (http://www.ethnologue.com). There is a great need for developing a text-to-speech system for Urdu because this population has low literacy rate and therefore speech interface would greatly assist in providing them access to information. One of the significant parts of a text-to-speech system is a natural language processor which takes textual input and converts it into an annotated phonetic string. To enable this, it is necessary to develop models which map textual input onto phonetic content. These models may be very complex for various languages having unpredictable behaviour (e.g. English), but Urdu shows a relatively regular behaviour and thus Urdu pronunciation may be modelled from Urdu text by defining fairly regular rules. These rules have been identified and explained in this paper.

1 Introduction

Text-to-speech synthesis is logically divided into two stages. The first stage takes raw text input, processes it and converts it into precise phonetic string to be spoken, appropriately annotated with prosodic markers (e.g. stress and intonation). The second stage takes this phonetic representation of speech and generates the appropriate digital signal using a particular synthesis technique. These stages may be referred to as Natural Language Processing (NLP) and Speech Synthesis (SS) respectively (e.g. Dutoit 1997, p.14).

For SS, formant based techniques (e.g. Klatt 1980) or diphone based techniques (e.g. Dutoit 1997) are normally employed and are generally script independent (as they are only dependent on temporal and spectral acoustic properties of the language and take input in script-neutral form, e.g. in IPA). However, NLP is very dependent on cultural and linguistic specific usage of script.

NLP may also be divided into further parts. The first component is dedicated to pre-processing, 'cleaning' and normalizing input text. Once the input text is normalized, the second component does phonological processing to generate a more precise phonetic string to be spoken. One of the first tasks in the Phonological Processing Component is to convert the input text into a phonemic string using Letter-to-Sound (LTS) rules. This string is then eventually converted to precise phonetic transcription after application of sound change rules and other annotations, as explained later. This paper overviews Urdu writing system, phonemic inventory, NLP for TTS and gives details of the LTS rules for Urdu (also see Rafique et at. (2001) and Hussain (1997: Appendix A), for introductory work).

2 Urdu Writing System and Phonemic Inventory

Urdu is written in Arabic script in Nastaleeq style using an extended Arabic character set. Nastaleeq is a cursive, context-sensitive and highly complex writing system (Hussain 2003). The character set includes basic and secondary letters, aerab (or diacritical marks), punctuation marks and special symbols (Hussain and Afzal 2001, Afzal and Hussain 2001). Urdu is normally written with only the letters. However, the letters represent just the consonantal content of the string and in some cases (under-specified) vocalic content. The vocalic content can be (optionally) completely specified by using the aerab with the letters. Aerab are normally not written and are assumed to be known by the native speaker, thus making it very hard for a foreigner to read. Certain aerab are also used to specify additional consonants. Urdu letters and aerab are given in Table 1 below.

چ	ج	ث	ٹ	ت	پ	ب	ا
ز	ڑ	ر	ذ	ڈ	د	خ	ح
ع	ظ	ط	ض	ص	ش	س	ژ
ن	م	ل	گ	ک	ق	ف	غ
			ے	ی	ء	ہ	و

ھ	ۃ	ں	آ

ـْ	ـّ	ـً	ـٰ	ـُ	ـِ	ـَ

Table 1: Urdu basic (top) and secondary (middle) letters and aerab (bottom)

Combination of these characters realizes a rich inventory of 44 consonants, 8 long oral vowels, 7 long nasal vowels, 3 short vowels and numerous diphthongs (e.g. Saleem et al. 2002, Hussain 1997; set of Urdu diphthongs is still under analysis). This phonemic inventory is given in Table 2.

The italicized phonemes, whose existence is still not determined, are not considered any further (see Saleem et al. 2002 for further discussion). Mapping of this phonetic inventory to the characters given in Table 1 is discussed later.

(a)

p	b	p^{h}	b^{h}	m	*m^{h}*	
t̪	d̪	t̪^{h}	d̪^{h}	n	*n^{h}*	
ʈ	ɖ	ʈ^{h}	ɖ^{h}			
k	g	k^{h}	g^{h}	ŋ	*ŋ^{h}*	
tʃ	dʒ	tʃ^{h}	dʒ^{h}	q	ʔ	
f	v	s	z			
ʃ	ʒ	x	ɣ	h		
r	*r^{h}*	ɽ	*ɽ^{h}*	j	l	*l^{h}*

(b)

i	e	ɛ	æ
u	o	ɔ	ɑ
ɪ	ʊ	ə	
ĩ	ẽ	æ̃	
ũ	õ	ɔ̃	ɑ̃

Table 2: Urdu (a) Consonantal and (b) Vocalic phonemic inventory

3 NLP for Urdu TTS

As discussed earlier, to enable text-to-speech system for any language, a Natural Language Processing component is required. The NLP system may have differing requirement for different languages. However, it always takes raw text input and always outputs precise phonetic transcription for a language. The system can be divided into two parts, Text-Normalization Component and Phonological Processing Component. These components may be further divided. A simplified schematic is shown in Figure 1[1].

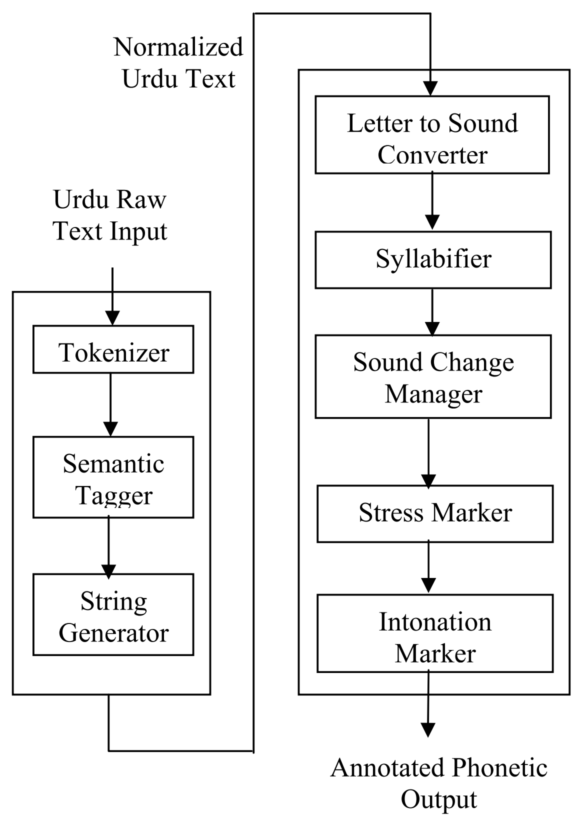


Figure 1: NLP architecture for Urdu TTS system

[1] This diagram is based on the architecture of Urdu Text to Speech system under development at Center for Research in Urdu Language Processing (www.crulp.org).

The Text Normalization component takes a character string as input and converts it into a string of letters. Within it, the Tokenizer uses the punctuation marks and space between words to mark token boundaries which are then stamped as words, punctuation, date, time and other relevant categories by the Semantic Tagger. The String Generator takes any non-letter based input (e.g. a number or a date containing digits) and converts it into a letter string.

After the input is converted into a string comprising only of letters, the Phonological Processing Component generates the corresponding phonetic transcription. This is done through a series of processes. The first process is to use Letter-to-Sound Converter (detailed below) to convert the normalized text input to a phonemic string. This process may also be referred to as grapheme-to-phoneme conversion. This is followed by Syllabifier, which marks syllable boundaries. The intermediate output is then forwarded to a module which applies Urdu sound change rules to generate the corresponding phonetic string. Following these modules, Stress Marker and Intonation Marker modules add stress and intonation to the string being processed. Re-syllabification is also performed after sound change rules are applied, in case phones are epenthesized or deleted and syllable boundaries require re-adjustment. Urdu shows a reasonably regular behavior and most of these tasks can be achieved through rule-based systems (e.g. see Hussain 1997 for stress assignment algorithm). This paper focuses on Letter-to-Sound rules for Urdu, the first in the series of modules in Phonological Processing Component.

4 Urdu Letter to Sound Rules

Urdu shows a very regular mapping from graphemes to phonemes. However, to explain the behavior, the letters need to be further classified into the following categories:

a. Consonantal characters
b. Dual (consonantal and vocalic) behavior characters
c. Vowel modifier character
d. Consonant modifier character
e. Composite (consonantal and vocalic) character

Similarly, the aerab set can also be divided into the following categories:

f. Basic vowel specifier
g. Extended vowel specifier
h. Consonantal gemination specifier
i. Dual (vocalic and consonantal) insertor

Finally, there is a third category which may take shape of an letter and aerab:

j. Vowel-aerab placeholder

The Consonantal characters in (a) above always represent a consonant of Urdu. In Urdu, there is always a single consonant corresponding to a single character of this category, unlike some other languages e.g. English maps "ph" string to phoneme /f/. Most of the Urdu consonantal characters fall into this category. These characters and corresponding consonantal phonemes are given in Table 3 below. A simple mapping rule would generate the phoneme corresponding to these characters.

چ	ج	ث	ٹ	ت	پ	ب
tʃ	dʒ	s	ʈ	t̪	p	b
ڑ	ر	ذ	ڈ	د	خ	ح
ɽ	r	z	ɖ	d̪	x	h
ط	ض	ص	ش	س	ژ	ز
t̪	z	s	ʃ	s	ʒ	z
گ	ک	ق	ف	غ	ع	ظ
g	k	q	f	ɣ	ʔ	z
		ۃ	ہ	ن	م	ل
		t̪	h	n	m	l

Table 3: Consonantal characters and their corresponding phonemes

Three characters of Urdu show dual behavior, i.e. in certain contexts they transform into consonants, but in certain other contexts, they transform into vowels. These characters are Alef (ا), vao (و), and Yay (ی or ے). Alef acts exceptionally in this category and therefore it is discussed separately in (j) below. Vao changes to /v/ and Yay changes to the approximant /j/ when they occur in consonantal positions (in onset or coda of a syllable). However, when they occur as nucleus of a syllable, they form long vowels. As an example, Yay occurs as a consonant when it occurs in the onset of single syllable word یار

(/jar/, “friend”) but is a vowel when it occurs word medially in بَیل (/bæl/, “ox”). These characters represent category (b) listed above.

There is only one character in category (c), the letter Noon Ghunna (ں), which does not add any additional sound to the string but only nasalizes the preceding vowel. This letter follows and combines with the category (b) characters (when occurring as vowels) to form the nasal long vowels, e.g. جا (/dʒɑ/, “go”) vs. جاں (/dʒɑ̃/, “life”). Catergory (d) is the letter Do-Chashmey Hay (ھ), which combines with all the stops and affricates to form aspirated (breathy or voiceless) consonants but does not add an additional phoneme. It may also combine with nasal stops and approximants to form their aspirated versions, though these sounds are not clearly established phonetically. As an example, adding this character adds aspiration to the phoneme /p/: پل(/pəl/, “moment”) vs. پھل (/pʰəl/, “fruit”). Finally, there is also a single character in category (e), the Alef Madda (آ). This character is a stylistic way of writing two Alefs and thus represents an Alef in consonantal position (see (j) below) and an Alef in vocalic position, forming /a/ vowel, e.g. اَب (/əb/, “now”) vs. آب (/ɑb/, “water”).

There are three Basic vowel aerab used in Urdu called Zabar (Arabic Fatha), Zer (Arabic Kasra) and Pesh (Arabic Damma). In addition, absence of these aerab also define certain vowels and thus this absence is referred to as Null aerab. They combine with characters to form vowels according to the following principles:

(i) *Short vowels*, when they occur with category (a) and (b) consonants not followed by category (b) letters.
(ii) *Long vowels*, when they occur with category (a) and (b) consonants followed and combined by category (b) characters.
(iii) *Long nasal vowels*, when they combine with category (a) and (b) consonants followed by category (b) characters followed by category (c) Noon Ghunna.

Different combination of these aerab with category (b) characters generate the various vowels, as indicated in Table 4 (all vowels shown in combination with ب (phoneme /b/) as a consonant character is required as a placeholder for the aerab).

Bay + Zabar	بَ	ə
Bay + Zer	بِ	ɪ
Bay + Pesh	بُ	ʊ
Bay + NULL + Alef	با	ɑ
Bay + NULL + Vao	بو	o
Bay + Zabar + Vao	بَو	ɔ
Bay + Pesh + Vao	بُو	u
Bay + NULL + Yay	بے	e
Bay + Zabar + Yay	بَے	æ
Bay + (NULL | Zer)[2] + Yay	بی	i
Bay + NULL + Alef + Noon Ghunna	باں	ɑ̃
Bay + NULL + Vao + Noon Ghunna	بوں	õ
Bay + Zabar + Vao + Noon Ghunna	بَوں	ɔ̃
Bay + Pesh + Vao + Noon Ghunna	بُوں	ũ
Bay + NULL + Yay + Noon Ghunna	بیں	ẽ
Bay + Zabar + Yay + Noon Ghunna	بَیں	æ̃

[2] NULL or Zer. It is controversial whether Zer is present for the representation of vowel /i/. One solution is to process both cases till the diction controversy is solved.

Bay + (Null | Zer) + Yay + Noon Ghunna (see Footnote 2)	بِیں	~ِ

Table 4: Letter and aerab combinations and corresponding vowels

Existence of the remaining vocalic phoneme /ɛ/ is controversial in Urdu as there is no way of expressing it using the Urdu writing system and because it is schwa conditioned by the following /h/ phoneme and only occurs in this context. However, it may exist phonetically e.g. in the word شہر (/ʃɛhɛr/, "city") (see discussion in Qureshi, 1992; also see some supporting acoustic evidence in Fatima et. al, 2003, e.g. duration of /ɛ/ is 136 ms compared with 235 ms for /æ/).

The next category (g) consists of Khari Zabar. This represents the vowel Alef and, whenever occurs on top of a Vao or Yay, replaces these sounds with the Alef vowel sound /a/ as in words زکٰوۃ (/zakat/,"zakat") and اعلیٰ (/əʔla/, special"). Sporadically Khari Zer and Ulta Pesh are referred to in Urdu as well but they generally do not occur on Urdu words. These are not considered here.

The gemination mark of category (h) is called Shad in Urdu and occurs on consonantal characters (of categories (a, b) except Alef). Shad geminates the consonant on which it occurs, which is normally word medially and inter-vocalically. As a result of gemination, the duplicate consonant acts as coda of previous syllable and onset of following syllable. For example, گدا (/gə.ḍa/, "a poor person") vs. گدّا (/gəḍ.ḍa/, "mattress").

The category (i) aerab, called Do-Zabar only occurs on Alef (in vocalic position) and converts the long vowel /a/ to short schwa followed by consonant /n/, e.g. in word فوراً (/fɔrən/, "immediately"). Do-Zer and Do-Pesh are similarly referred to in Urdu but are not generatively used and are mostly in foreign words especially of Arabic and are not considered further here. If considered, they would present a similar analysis. Finally, (j) is a very interesting category as it represents allo-graphs Alef and Hamza (former a character and latter (arguably) an aerab and character[3]). Both of them are default markers and occur in complimentary distribution, Alef always word initially and Hamza always otherwise. As discussed earlier, aerab in Urdu always need a *Kursi* ("seat"). If a short vowel occurs word initially without a consonant (i.e. in a syllable which has no onset), there is no placeholder for aerab. A default place holder is necessary and Alef is used. Word medially, if there is an onset-less syllable, Urdu faces the same problem. In these cases, Hamza (instead of Alef) is used as a placeholder for aerab. There are two further possible sub-cases. In one, the preceding syllable is open and ends with a vowel. This case is very frequent and Hamza is introduced inter.-vocalically (e.g. فائِدہ /fa.ɪdɛh/, "advantage"). In the second less productive sub-case, the preceding syllable is closed by a coda consonant. In this case, Hamza is (optionally) used with Alef (e.g. both forms are correct: جرأت / جرات /dʒʊr.ət/, "courage"). Hindi which employs a different mechanism by defining different shapes for vowels word-initially and word-medially (Matras). The Matras are anchored onto the consonants, e.g. in आने वाला , "about to come" vowel /a/ is written as आ word initially, but is written as ा word medially).

These rules have been implemented in an on-going project (see Footnote 1 above) and are successfully generating the desired phonemic output. This phonemic output is passed through sound change rule module to generate the desired phonetic form.

5 Conclusion

This paper briefly discusses the architecture of Natural Language Processing portion of an Urdu Text-to-Speech system. It explains the details of Urdu consonantal and vocalic system and Urdu letters. Urdu shows regular behavior and thus the phonemic forms are predictable from the textual input. The letter-to-sound rules define this

[3] Hamza sometimes requires a *Kursi* or seat (قائل and not قأل) and sometimes does not (پلاؤ and not پلائو) indicating it behaves both like a character and an aerab. It is still unclear on how this behavior is distributed and whether it is predictable. As it is a script centric issue, it is not discussed further here.

mapping and are thus essential for developing Urdu TTS.

6 Acknowledgements

This work has been partially supported by the grant for "Urdu Localization Project: MT, TTS and Lexicon" by E-Government Directorate of Ministry of IT and Telecommunications, Government of Pakistan.

The author also wishes to thank anonymous reviewers for comments, especially on glottal stop and Hamza and Tahira Khizar and Qasim Vaince for eventual discussion on the role of Hamza in Urdu script.

References

M. Afzal and S. Hussain. 2001. Urdu Computing Standards: Development of Urdu Zabta Takhti (UZT 1.01). *Proceedings of IEEE International Multi-topic Conference,* Lahore, Pakistan.

T. Dutoit. 1997. *An Introduction to Text-to-Speech Síntesis.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

N. Fatima and R. Aden. Vowel Structure of Urdu. 2003. *CRULP Annual Student Report* published in *Akhbar-e-Urdu,* April-May, National Language Authority, Islamabad, Pakistan.

S. Hussain. 2003. www.LICT4D.aisa/Fonts/ Nafees_Nastalique. *Proceedings of 12th AMIC Annual Conference on E-Worlds: Governments, Business and Civil Society*, Asian Media Information Center, Singapore.

S. Hussain. 1997. *Phonetic Correlates of Lexical Stress in Urdu.* Unpublished Doctoral Dissertation, Northwestern University, Evanston, USA.

S. Hussain, and M. Afzal. 2001. Urdu Computing Standards: Urdu Zabta Takhti (UZT 1.01). *Proceedings of IEEE International Multi-topic Conference,* Lahore, Pakistan.

D. H. Klatt. 1980. Software for Cascade/Parallel Formant SynthesiZer. *JASA 67*: 971-995.

M. M. Rafique, M. K. Riaz, and S.R. Shahid. 2002. Vowel Insertion Grammar. *CRULP Annual Student Report* published in *Akhbar-e-Urdu,* April-May, National Language Authority, Islamabad, Pakistan.

B. A. Qureshi. 1992. *Standard Twentieth Centuary Dictionary: Urdu to English.* Educational Publishing House, New Dehli, India.

A. M. Saleem, H. Kabir, M.K. Riaz, M.M. Rafique, N. Khalid, and S.R. Shahid. 2002. Urdu Consonantal and Vocalic Sounds. *CRULP Annual Student Report* published in *Akhbar-e-Urdu,* April-May, National Language Authority, Islamabad, Pakistan.

Urdu Localization Project: Lexicon, MT and TTS (ULP)

Sarmad HUSSAIN
Center for Research in Urdu Language Processing,
National University of Computer and Emerging Sciences
B Block, Faisal Town
Lahore, Pakistan
sarmad.hussain@nu.edu.pk

Abstract

Pakistan has a population of 140 million speaking more than 56 different languages. Urdu is the lingua franca of these people, as many speak Urdu as a second language, also the national language of Pakistan. Being a developing population, Pakistani people need access to information. Most of the information over the ICT infrastructure is only available in English and only 5-10% of these people are familiar with English. Therefore, Government of Pakistan has embarked on a project which will generate software to automatically translate the information available in English to Urdu. The project will also be able to convert Urdu text to speech to extend this information to the illiterate population as well. This paper overviews the overall architecture of the project and provides briefs on the three components of this project, namely Urdu Lexicon, English to Urdu Machine Translation System and Urdu Text to Speech System.

1 Introduction

In today's information age it is critical to provide access to information to people for their development. One precursor to this access is availability of information in the native languages. Due to limitations in technology, it has not been possible to generate information in many languages of the world. However, with recent advances in internationalization and localization technology, many languages are not enabled. However, as this is recent development, the published content in these languages is still limited, and far lags behind the content available for English, Spanish and some other languages spoken in developed countries. Realizing this gap in content and the need to provide access to information to its citizens, Government of Pakistan has recently launched Urdu Localization Project[1]. This project will enable translation and access of English content to literate and illiterate Urdu speakers.

Urdu Localization Project aims to provide access to existing English language content to Urdu language speakers. The project has three components: Urdu Computational Lexicon, English-to-Urdu Machine Translation System, Urdu Text-to-Speech system. This paper briefly describes the architecture and work achieved to-date for different systems within ULP.

2 ULP Architecture

As indicated, ULP comprises of three largely independent systems: Lexicon, MT and TTS, though these components may also be integrated to develop a written and oral interface to information. The project has three architectural layers. At the base are the core data and engines for Lexicon, MT and TTS. The middle layer provides public programming interfaces to these engines (APIs) so that they may be integrated with end-user applications at the top layer or used by third-party applications. Both the engine and API layer components are being developed in standard C/C++ to enable them to compile on all platforms (e.g. Microsoft, Linux, Unix). The user-end/top layer has to be technology centric and is currently being enabled in Microsoft platform. The lexicon will be given a web interface for user access. In addition, plug-ins for internet and email clients will be developed for MT and TTS to enable end-users to translate and re-display English websites in Urdu and also enable them to convert the translated Urdu text to speech. This is shown in Figure 1 below. In the figure the layers and systems are demarcated (horizontally and vertically respectively). The figure also shows that MT and TTS may be using the Lexicon through the APIs for getting appropriate data.

[1] Urdu Localization Project is a three-year initiative being undertaken by Center for Research in Urdu Language Processing (www.crulp.org) and is funded through a grant by E-Government Directorate of Ministry of IT&Telecom., Government of Pakistan.

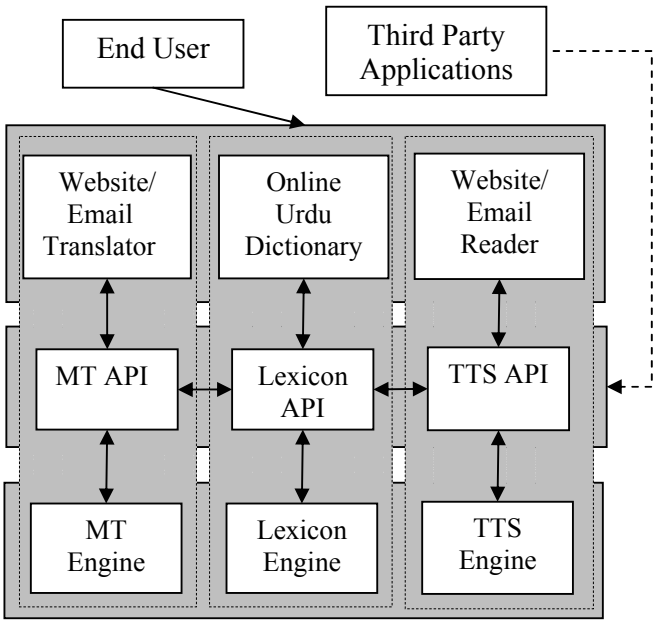


Figure 1: Architecture Diagram for ULP

These three systems are discussed briefly below.

2.1 Urdu Lexicon

Urdu Computational Lexicon being designed would be holding more than 25 dimensions of a single headword. The first task to date has been to determine this hierarchical storage structure. The structure required for end-user has been finalized. However, requirements for computational applications, e.g. MT, are still being finalized. This was perhaps one of the most challenging tasks as there are currently no standards which exist, although some guidelines are available. In addition, Urdu also had some additional requirements (e.g. multiple plural forms, depending on whether the word is derived from Arabic or Sanskrit). Entries of more than thirty thousand headwords and complete entry of about a thousand headwords along with specification of at least 15 entries has already been done. Currently more content is being generated. In addition, work is under progress to define the physical structure of the lexicon (e.g. storage and retrieval models). The prototype showing this application is also available in Microsoft platform.

2.2 English-Urdu Machine Translation

Work is under progress to develop English to Urdu MT engine. The translation is based on LFG formalism and is developing grammars, lexica and the parsing/mapping/generation engine for LFG. Mapping and Generation prototypes have already been developed and are integrated with a freely available LFG parser for internal testing. In addition sample grammars for English, Urdu and English-Urdu mapping have also been written. The prototype covers about 10 percent of grammatical rules and already translates within the limited vocabulary of the engine. The work is being extended to write the parser and rewrite mapper and generator and to develop English, Urdu and English Urdu grammars and lexica.

2.3 Urdu Text to Speech System

The Urdu TTS is divided into two main part, the Urdu Natural Language Processor and Urdu Speech Synthesizer. The work on NLP is completed (except the intonational module, on which preliminary work has been completed). The NLP processor inputs Urdu Unicode text and output narrow phonetic transcription with syllable and stress markers. The NLP processor is integrated with Festival speech synthesis system (though by-passes its NLP module). A vocabulary of about 500 words is already defined at the diphones have been created. Prototype application is already developed which synthesized these single words. Work is currently in progress to define Urdu intonational and durational model. In addition, work is also under progress to extend the vocabulary and functionality to synthesize complete sentences. The functional prototype works on both Linux an Microsoft platforms.

3 Conclusion

Most of the work being done in the project is novel. Urdu language is not very well defined for use with computers. Script, speech and language aspects of Urdu are being studied, documented and implemented in this project. The project is also testing the work which has been matured on western languages but only being recently exposed to other languages, e.g. the lexical recommendations by ISLE, LFG framework, use of LFG for MT, speech modeling of Urdu (both spectral and temporal) and more. Non-functional issues including performance are also being negotiated. Pre-compiled lexica, user-centric pre-stored performance-enhancing profiles and frequency lists, etc. are part of the architectural tasks being addressed. Though only initial work has been done, this work in itself is substantial, and has raised many questions which will be answered as the project progresses.

FarsiSum - A Persian text summarizer

Martin Hassel
KTH NADA
Royal Institute of Technology
100 44 Stockholm, Sweden
xmartin@nada.kth.se

Nima Mazdak
Department of Linguistics
Stockholm University
106 91 Stockholm, Sweden
nima.mazdak@comhem.se

Abstract

FarsiSum is an attempt to create an automatic text summarization system for Persian. The system is implemented as a HTTP client/server application written in Perl. It uses modules implemented in an existing summarizer geared towards the Germanic languages, a Persian stop-list in Unicode format and a small set of heuristic rules.

1 Introduction

FarsiSum is an attempt to create an automatic text summarization system for Persian (Mazdak, 2004). The system is implemented as a HTTP client/server application written in Perl. It uses modules implemented in SweSum (Dalianis 2000), a Persian stop-list in Unicode format and a small set of heuristic rules. The stop-list is a file including the most common verbs, pronouns, adverbs, conjunctions, prepositions and articles in Persian. The words not included in the stop-list are supposed to be nouns or adjectives. The idea is that nouns and adjectives are meaning-carrying words and should be regarded as keywords.

The current implementation of FarsiSum is still a prototype. It uses a very simple stop-list in order to filter and identify the important keywords in the text. Persian acronyms and abbreviations are not detected by the current tokenizer.

In addition, Persian syntax is quite ambiguous in its written form (Megerdoomian and Rémi 2000), which raises certain difficulties in automatic parsing of written text and automatic text summarization for Persian.

For example, selection of important keywords in the *topic identification* process will be affected by the following word boundary ambiguities:

- Compound words may appear as two different words.
- Bound morphemes may appear as free morphemes or vice versa.

These ambiguities are not resolved in the current implementation.

2 SweSum

SweSum[1] (Dalianis 2000) is a web-based automatic text summarizer developed at the Royal Institute of Technology (KTH) in Sweden. It uses text extraction based on statistical and linguistic as well as heuristic methods to obtain text summarization and its main domain is Swedish HTML-tagged newspaper text[2].

2.1 SweSum's architecture

SweSum is a client/server application. The summarizer is located on the web server. It takes a Swedish text as input and performs summarization in three phases to create the final output (the summarized text).

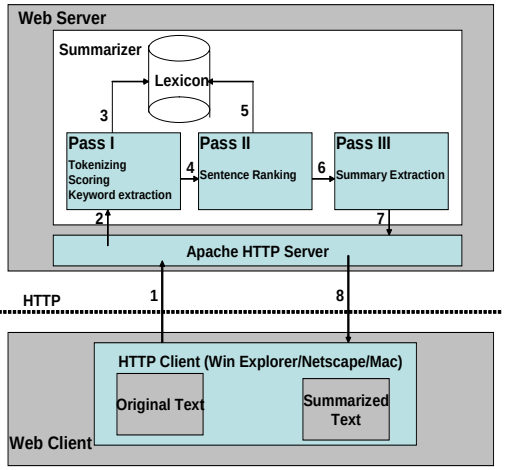


Figure 1: SweSum architecture

Pass 1: The sentence and word boundaries are identified by searching for periods, exclamation and question marks etc (with the exception of when periods occur in known abbreviations). The sentences are then scored by using statistical, linguistic and heuristic methods. The scoring depends on, for example, the position of the sentence in the text, numerical values in and

[1] An online demo is available at http://swesum.nada.kth.se/index.html

[2] SweSum is also available for English, Danish, Norwegian, Spanish, French, German, and now with the implementation described in this paper, Farsi.

various formatting of the sentence such as bold, headings, etc.

Pass 2: In the second pass, the score of each word in the sentence is calculated and added to the sentence score. Sentences containing common content words get higher scores.

Pass 3: In the third pass, the final summary file (HTML format) is created. This file includes:

- The highest ranking sentences up to a pre-set threshold.
- Optionally, statistical information about the summary, i.e. the number of words, number of lines, the most frequent keywords, actual compression rate etc.

For most languages SweSum uses a static lexicon containing many high frequent open class words. The lexicon is a data structure for storing key/value pairs where the key is the inflected word and the value is the stem/root of the word. For example *boy* and *boy**s*** have different inflections but the same root (lemma).

3 FarsiSum

FarsiSum is a web-based text summarizer for Persian based upon SweSum. It summarizes Persian newspaper text/HTML in Unicode format. FarsiSum uses the same structure used by SweSum (see Figure 2), with exception of the lexicons, but some modifications have been made in SweSum in order to support Persian texts in Unicode format.

3.1 User Interface

The user interface includes:

- The first page of FarsiSum on WWW presented in Persian[3].
- A Persian online editor for writing in Persian.

The final summary including statistical information to the user, presented in Persian.

3.2 Stop List

The current implementation uses a simple stop list rather than a full-fledged Persian lexicon. The stop-list is a HTML file (UTF-8 encoding) containing about 200 high-frequency Persian words including the most common verbs, pronouns, adverbs, conjunctions, prepositions and articles.

The stop-list has been successively built during the implementation phase by iteratively running FarsiSum in order to find the most common words in Persian.

The assumption is that words not included in the stop-list are nouns or adjectives (content words) and should be counted as such in the word frequency list.

3.3 Tokenizer

The tokenizer is modified in order to recognize Persian comma, semi colon and question mark.

- Sentence boundaries are found by searching for periods, exclamation and question marks as well as
 (the HTML new line) and the Persian question mark (؟).
- The tokenizer finds the word boundaries by searching for characters such as "**.**", "**,**", "**!**", "**?**", "**<**", "**>**", "**:**", spaces, tabs and new lines. Persian semi colon, comma and question mark can also be recognized.
- All words in the document are converted from ASCII to UTF-8. These words are then compared with the words in the stop-list. Words not included in the stop list are regarded as content words and will be counted as keywords.

The word order in Persian is SOV[4], i.e. the last word in a sentence is a verb. This knowledge is used to prevent verbs from being stored in the *Word frequency table*.

3.4 Architecture

FarsiSum is implemented as a HTTP client/server application as shown in Figure 2. The summarization program is located on the server side and the client is a browser such as Internet Explorer or Netscape Navigator.

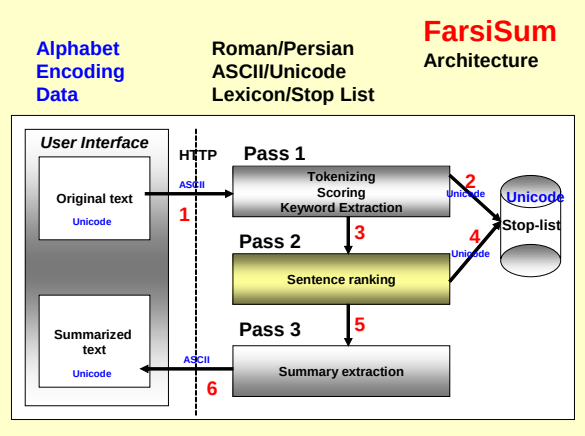


Figure 1: FarsiSum architecture

[3] http://www.nada.kth.se/iplab/hlt/farsisum/index-farsi.html

[4] **SOV** stands for **S**ubject, **O**bject and **V**erb.

The summarization process starts when the user (client) clicks on a hyperlink (*summarize*) on the FarsiSum Web site:

- The browser (Web client) sends a summarization request (marked 1 in Figure 2) to the Web server where FarsiSum is located. The document/ (URL of the document) to be summarized is attached to the request. (The original text is in Unicode format).
- The document is summarized in three phases including tokenizing, scoring and keyword extraction. Words in the document are converted from ASCII to UTF-8. These words are then compared with the words in the stop-list (2-5).
- The summary is returned back to the HTTP server that returns the summarized document to the client (6).

The browser then renders the summarized text to the screen.

4 Conclusions

The system would most certainly benefit from deeper language specific analysis, but with no access to Persian resources, in this system fairly language independent methods have proven to come a long way.

References

Dalianis, H. 2000. *SweSum - A Text Summarizer for Swedish*, Technical report, TRITA-NA-P0015, IPLab-174, NADA, KTH, October 2000.

Mazdak, N. 2004. *FarsiSum - a Persian text summarizer*, Master thesis, Department of Linguistics, Stockholm University, (PDF)

Megerdoomian, Karine and Rémi, Zajac 2000. *Processing Persian Text: Tokenization in the Shiraz Project*. NMSU, CRL, Memoranda in Computer and Cognitive Science (MCCS-00-322).

Stemming the Qur'an

Naglaa Thabet
School of English Literature, Language and Linguistics
University of Newcastle
Newcastle upon Tyne, UK, NE1 7RU
n.a.thabet@ncl.ac.uk

Abstract

In natural language, a stem is the morphological base of a word to which affixes can be attached to form derivatives. Stemming is a process of assigning morphological variants of words to equivalence classes such that each class corresponds to a single stem. Different stemmers have been developed for a wide range of languages and for a variety of purposes. Arabic, a highly inflected language with complex orthography, requires good stemming for effective text analysis. Preliminary investigation indicates that existing approaches to Arabic stemming fail to provide effective and accurate equivalence classes when applied to a text like the Qur'an written in Classical Arabic. Therefore, I propose a new stemming approach based on a light stemming technique that uses a transliterated version of the Qur'an in western script.

1 Introduction

Stemming has been widely used in several fields of natural language processing such as data mining, information retrieval, and multivariate analysis. Some applications of multivariate analysis of text involve the identification of lexical occurrences of word stems in a text. Such lexical analysis, in which the frequency of word occurrences is significant, cannot be done without some form of stemming.

In morphology, variants of words which have similar semantic interpretations are considered to belong to the same stem and to be equivalent for purposes of text analysis and information retrieval. For this reason, a number of stemming algorithms have been developed in an attempt to reduce such morphological variants of words to their common stem.

Various stemming algorithms for a number of languages have been proposed. The structure of these stemmers range from the simplest technique, such as removing suffixes, to a more complicated design which uses the morphological structure of words to derive a stem.

In case of Arabic, several stemming algorithms have been developed. The major inadequacy of existing systems to stem the Qur'an results from the fact that most of them deal with Modern Standard Arabic as their input text; the language of the Qur'an is Classical Arabic. Orthographic variations and the use of diacritics and glyphs in the representation of the language of Classical Arabic increase the difficulty of stemming. In many respects, the Qur'an, with its unique lexicon and orthography requires dedicated attention.

Therefore, I have developed a new light stemmer that uses the Qur'an in western transliteration to improve the effectiveness of the stemming of the text.

2 Stemming in Arabic

Arabic belongs to the Semitic family of languages, and as such differs from European languages morphologically, syntactically and semantically. The Arabic language is somewhat difficult to deal with due to its orthographic variations and its complex morphological structure. Xu et al. provide an overview of the challenges the Arabic language creates for information retrieval [10, 11].

2.1 Arabic Morphology

The grammatical system of the Arabic language is based on a root-and-affix structure and is considered as a root-based language. Most Arabic words are morphologically derived from a list of roots, to which many affixes can be attached to form surface words. Most of these roots are made up of three consonants which convey semantics. In addition to the different forms of the Arabic word that results from the derivational and inflectional process, most prepositions, conjunctions, pronouns, and possession forms are attached to the Arabic surface form.

2.2 Arabic Orthography

Orthographic variations are prevalent in Arabic. Vocalized texts make use of diacritics to represent short vowels. The omission of such diacritics in non-vocalized text gives rise to ambiguity, specifically if words are read out of context. Other spelling variations include changing the letter ي to

ى at the end of a word and replacing إ, أ, and آ with plain ا. A sense of discrimination and a good knowledge of grammar and usage are required if one is to avoid misreading a word.

In terms of multivariate analysis of text as well as information retrieval, the combination of a rich morphology and a pervasively ambiguous writing system results in a degree of complexity such that some sort of pre-processing and classification is required. Therefore, stemming is very important for Arabic text analysis.

2.3 Approaches to Arabic Stemming

Several stemming algorithms for Arabic have been proposed based on different principles; each produces rather different sets of stem classifications. It is possible to evaluate these stemming algorithms by the accuracy of the results they produce. Larkey et al. gives a good summary of stemming approaches for the Arabic language [9]. The most common approaches used in Arabic stemming are the light and the root-based stemmers.

Root-based Stemming is based on removing all attached prefixes and suffixes in an attempt to extract the root of a given Arabic surface word. Several morphological analyzers have been developed, e.g. Buckwalter [3], Khoja and Garside [7] and Darwish [5].

Light Stemming is used not to produce the linguistic root of a given Arabic surface form, but to remove the most frequent suffixes and prefixes. The most common suffixation includes duals and plurals for masculine and feminine, possessive forms, definite articles, and pronouns. Several light stemmers have been developed, all based on suffix and prefix removal and normalization. Examples of light stemmers include: Aljlayl & Frieder's Stemmer [2], Darwish's Al-Stem [6], Chen & Gey's TREC 2002 Stemmer [4], and Larkey et al.'s U Mass Stemmer [8, 9].

All light stemmers adhere to the same steps of normalization and stemming. The main difference among them is the number of prefixes and suffixes removed from each one. During the normalization process, all diacritics, punctuation, and glyphs are removed. The light stemmers had different stopword lists consisting of Arabic pronouns, particles and the like removed after minimal normalization. Test results of previous researchers as in [2, 8], proved that the light stemmer achieved superior performance over the root-based approach since it reduces sense ambiguity by grouping semantically related words into the same class.

Although light stemming can correctly classify many variants of words into large stem classes, it can fail to classify other forms that should go together. For example, broken plurals for nouns and adjectives do not get conflated with their singular forms, and past tense verbs do not get conflated with their present tense forms, because they retain some affixes and internal differences.

3 Stemming the Qur'an

My main objective for stemming the Qur'an is to prepare the text as data for multivariate analysis of the lexical semantics of the Qur'an using self-organizing maps in which words with similar meanings are placed at the same or neighbouring points so that the topological relations among them represent degrees of semantic similarity. This work requires the construction of vector space models of the suras (chapters) of the Qur'an such that each sura is represented by a vector indicating the occurrence frequency of variables. This involves counting the occurrences of lexical items in the Qur'an. Such a task cannot be done accurately without some sort of stemming of words in the text.

The Qur'an has two significant textual features. The first is that the Classical Arabic language in which the Qur'an is written has created difficulty in reading and understanding it, even for the Arabs themselves. Its lexicon, morphology and grammar are more complicated than Modern Standard Arabic. It, therefore, requires specific attention.

The second significant point is the wide use of vocalization. Diacritics (ْ, ّ, ٍ, ُ, َ, ِ, ٌ, ً) representing short vowels are prevalent in the Qur'an. Every word, even every letter is marked with a diacritic. The meanings of the words in the Qur'an require the use of such diacritical marks; otherwise it becomes very difficult to comprehend their meanings especially when out of context.

Vocalized text, in Arabic includes diacritics for short vowels and other details. Thus, a word could have several meanings when marked with different diacritics. (see Table 1).

Word	Transliteration	Meaning
مُلْك	mulk	reign
مَلِك	malik	king
مَلَك	malak	angel
خُلُق	khuluq	morals
خَلْق	khalq	creation
اَمَة	amah	female slave
اُمَّة	ummah	nation

Table 1. Orthographic variations of words

For those reasons stemming the Qur'an is not an easy task. In principal, the way existing Arabic stemmers are structured indicates that they will not work reliably on the stemming of the Qur'an. Most of the existing stemmers rely on Modern Standard Arabic as their input script. This modern form of

Arabic is a simplified form of Classical Arabic. The main differences between both forms are that Modern Standard Arabic has less orthographic variation, a less complicated lexicon and a more modern vocabulary. The following two points are also significant regarding the use of existing stemmers to stem the Qur'an.

First, the root-based algorithm increases word ambiguity. The root algorithm stems the surface form to a base form from which the word variants are derived. A major problem with this type of stemmer is that many word variants are different in meaning, though they originate from one identical root. For example words like *hasib* (he thought), *hasaba* (he counted), and *hasab* (of noble origin) are all derived from the same root *hsb.* Therefore, the over-stemming of the root algorithm results in the deterioration of the retrieval performance as compared to the light stemming algorithm. As noted by Khoja [7], another problem that the stemmer faces is that some of the letters that appear to be affixes are in fact part of the word.

Second, the light stemmers perform better than the root-based algorithms, though not entirely efficiently. All initial steps of the light-based algorithms require normalization which involves the removal of diacritics. Thus, if diacritics were removed from the words listed in Table 1 above, there would be no other way to indicate the difference in meaning of all word variants. The normalization technique, though it appears simple, increases ambiguity. If normalization was applied to the Qur'an, it would leave the text highly ambiguous. As the case with root-based algorithms, some of the suffixes and prefixes to be removed using light stemmers are originally part of the word.

Therefore, I propose a new light stemming approach that gives better results, particularly when applied to a rich vocalized text as the Qur'an. The stemmer is basically a light stemmer to remove prefixes and suffixes and is applied to a version of the Qur'an transliterated into western script.

The use of the transliteration is highly significant for resolving the problem of diacritics in the Qur'an. Given that the transliteration of the Qur'an is available in western script, the problem of diacritics is resolved, since in the transliterated version of the Qur'an, each diacritic is translated into a letter in Roman script. Thus, the ambiguity that arises when removing the diacritics from the Arabic text is avoided. So, while the word ملك could have three different meanings when it appears without diacritics in Arabic, in transliteration each meaningful word has a single representation. (see Table 1).

Another advantage of using transliteration is avoiding the removal of suffixes and prefixes that sometimes could be part of the word. The prefix بـ (pronounced as "bi") is very common in Arabic. This preposition resembles the letter ب of the Arabic alphabet. Thus, removing this letter indistinguishably would cause ambiguity if the letter is part of a word. For example, in words as بَحر (sea), بُرهان (proof), the letter بـ is part of the word, whereas, in بِقلم (with a pen) the بـ is a preposition. If the diacritics that are marking the letter ب were removed, the first letter in each word would be exactly the same, though different in pronunciation. Therfore, stemming the words from the prefix بـ, in general, would be incorrect. When transliterating the same three words (بَحر, بُرهان, بِقلم) the prefix بـ would be represented as ba (bahr), bu (burhan), and bi (biqalam) respectively. The proposed light stemmer would only include "bi" as a prefix thus, avoiding removing the other representations of that letter. A few stems in Arabic begin with "bi"; those are added to a stopword list to be removed before stemming. The same process would be applied to the other prefixes to be removed such as (لـ, كـ, فـ, سـ, ال), (la/li, ka, fa, sa, al).

3.1 Implementation

The stemmer has been developed for the windows environment in Delphi, an object-oriented programming language which creates a graphical user interface to facilitate the presentation of its applications.

a. Preprocessing

Rather than the use of Arabic script, the system uses a Roman transliteration of the Qur'an which is formatted on the Web as HTML. This presents a particular problem that need to be remedied before the text can be stemmed. The problem is that some phonemically important distinctions, i.e., distinctions that are represented by different graphs in Arabic, are shown using HTML tags; when the HTML files are saved as text, these tags disappear, and the distinctions are lost. The Arabic phonemes (آ, ط, ح, ص, ض, ذ, ظ) are represented in the HTML transliteration files as underlined (a, t, h, s, d, th, *th*) respectively.

Preprocessing involves (1) stripping out the entire HTML markup, and (2) before doing so, replacing all the above phonemes with the following characters: a^, t^, h^, s^, d^, z^, z*. The result is a pure text file in ASCII codes.

b. Construction of stopword list

A stopword list of all the words to be excluded from the stemming process was compiled. The list was manually constructed using a concordance of the Qur'anic lexicon compiled by Abd Al-Baqi [1]. It consists of words which begin with the same letters which compose Arabic prefixes. Arabic pronouns, prepositions and names of people and places were also included in the stopword list.

c. Construction of stemmer

The algorithm for the stemmer is as follows:

Step 1. Prefix Stemming

The program reads individual suras from text files, replaces all uppercase letters with lower case letters and constructs a list of word lists, where each word list contains all the words in a single sura. It then reads single words from each word list and compares the current word supplied as a parameter to each successive word in the stopword list. If the word is found in the stopword list, it is excluded from prefix stemming; otherwise it adheres to following procedures:

- Remove prefixes (wa, fa, la, li, lil, bi, ka, sa, s^a, al)
- After stemming, the word is inserted back into the word list.

Step 2. Suffix Stemming

Six groups of suffixes are identified ranging from one-letter suffixes to six-letter suffixes. The system starts stemming the words in the word lists from the longest prefixes (six-letter prefixes) to the three-letter prefixes. Stemming the one and two-letter suffixes causes some ambiguity, since some of the suffixes could sometimes be part of the word stem. To resolve this problem, the stemmer sorts the words alphabetically. In the sorted list of words, if a given sequence displays a variety of suffixes including one and two-letter suffixes, the suffixes are removed and the stem is retained, otherwise the word is left intact.

3.2 Results

Preliminary results for seven long suras selected randomly and representing 6% of the Qur'an show that the stemmer achieves an accuracy of 99.6% for prefix stemming and 97% for suffix stemming. As the stemmer is being used, some inaccuracies were detected, but investigation shows that they are mainly to do with erroneous lexical items in the transliterated Qur'an. An evaluation of the system with accuracy figures should be available shortly for the entire Qur'anic text.

4 Conclusion

Stemming is important for a highly inflected language as Arabic. Existing Arabic stemmers, though produced effective results in some applications, failed to provide good stemming for the Qur'an. Therefore, I have proposed this new method of using transliterated script, which gave good preliminary results. Ongoing work on the system is focused on improving the accuracy of the results either by modifying the algorithms or editing the transliteration of the Qur'an.

References

[1] M.F. Abd Al-Baqi. 1987. *Al-Ma&jam Al-Mufahras li-alfaz Al-Qur'an Al-Karim.* Dar Al-hadith, Cairo.

[2] M. Aljlayl and O. Frieder. 2002. On Arabic Search: Improving the retrieval effectiveness via a light stemming approach. In *Proceedings of CIKM'02*, VA, USA.

[3] T. Buckwalter. 2002. *Buckwalter Arabic Morphological Analyzer Version 1.0*. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49.

[4] A. Chen and F. Gey. 2002. Building an Arabic stemmer for information retrieval. In *Proceedings of TREC 2002,* Gaithersburg, Maryland.

[5] K. Darwish. *An Arabic Morphological analyzer*. http://www.glue.umd.edu/~Kareem/research/

[6] K. Darwish and D. Oard. 2002. CLIR Experiments at Maryland for TREC-2002: Evidence combination for Arabic-English Retrieval. In *Proceedings of TREC 2002,* Gaithersburg, Maryland.

[7] S. Khoja and R. Garside. 1999. *Stemming Arabic text.* Computing Department, Lancaster University, Lancaster. http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps

[8] L. S. Larkey and M. E. Connell. 2001. Arabic information retrieval at UMass. In *Proceedings of TREC 2001*, Gaithersburg: NIST, 2001.

[9] L. S. Larkey, L. Ballesteros and M.E.Connell. 2002. Improving stemming for Arabic information retrieval: Light Stemming and co-occurrence analysis. In *SIGIR 2002*, Tampere, Finland: ACM, 2002.

[10] J. Xu, A. Fraser and R. Weischedel. 2001. TREC 2001 cross-lingual retrieval at BBN. In *TREC 2001*, Gaithersburg: NIST, 2001.

[11] J. Xu, A. Fraser and R. Weischedel. 2002. Empirical studies in strategies for Arabic information retrieval. In *SIGIR 2002*, Tampere, Finland: ACM, 2002.

Language Weaver Arabic->English MT

Daniel MARCU, Alex FRASER, William WONG, Kevin KNIGHT
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA, USA, 90292
{marcu,afraser,wong,knight}@languageweaver.com

Abstract

This presentation is primarily a demonstration of a working statistical machine translation system which translates Modern Standard Arabic into English.

1 Overview

Language Weaver has produced a high-performance statistical Arabic-to-English machine translation system, based on research work conducted at the University of Southern California, Information Sciences Institute (USC/ISI). Getting resource-unlimited laboratory systems to run in real time, on a typical desktop Windows machine, is among Language Weaver's contributions. The system is designed to provide broad general coverage of Arabic news, and is currently used at various sites within the U.S. Government.

The Arabic->English translation system to be demonstrated has been prepared in versions that require 1 or 2 GB of RAM, and run on a 1.5GHz or faster processor and translates at a minimum rate of 500 words per minute. The system includes an option to trade off speed for quality in the translation process allowing users to select the fastest possible gisting-quality output, or the best possible translation quality for each sentence.

2 Demonstration

The translation system will be demonstrated on current news, and possibly other postings from Internet, or other files: