JEP-TALN 2004, Session Traitement Automatique de l'Arabe, Fès, 20 avril 2004

Large Scale Lexeme Based Arabic Morphological Generation

Nizar Habash
University of Maryland Institute for Advanced Computer Studies
University of Maryland College Park
College Park, Maryland, 20742 USA
habash@umiacs.umd.edu

Mots-clefs – Keywords

morphologie arabe, génération automatique du langauge naturel
Arabic Morphology, Natural Language Generation

Résumé - Abstract

Cet article présente Aragen, un système de génération morphologique pour l'arabe à large couverture basé sur la notion de lexème. Aragen utilise les bases de données de l'analyseur morphologique de Buckwalter, avec un algorithme adapté pour la génération. La construction d'Aragen a aussi nécessité l'extension de la base de données morphologiques pour aboutir à une représentation basée sur la notion de lexème et de traits. L'évaluation de la couverture d'Aragen montre qu'Aragen produit des résultats significativement meilleurs qu'un système de comparaison simple.

This paper presents Aragen, a large-scale lexeme-based Arabic morphological generation system. Aragen uses the databases of the Buckwalter Arabic morphological analyzer with a different engine focused on generation rather than analysis. The building of Aragen involved the reversal of the Buckwalter analyzer and extending its databases to be used in a lexeme-plus-feature level of representation. An evaluation of Aragen in terms of Overgeneration / Undergeneration Error Rates indicates it to be more than one order of magnitude better than a simple baseline.

1 Introduction and Background

Arabic morphological analysis has been the focus of researchers in natural language processing for a long time. This is due to challenging features of Arabic morphology such as ambiguity resulting from optional diacritization, infix morphemes, and root-and-pattern morphology. Numerous morphological analyzers have been built for a wide range of application areas from Information Retrieval (IR) to Machine Translation (MT) and Natural Language Generation (NLG) in a variety of linguistic theoretical contexts (Kiraz1994; Beesley1996; Buckwalter2002; Darwish2002; Aljlayl and Frieder2002).

Arabic morphological generation, by comparison, has received little attention although the types of generation problems can be as complex as in analysis. Finite-state transducer (FST) approaches to morphology (Koskenniemi1983) and their extensions for Arabic such as Xerox Arabic analyzer (Beesley1996) are attractive for being generative models. However, a major hurdle to their usability is that lexical and surface levels are very close (Karttunen et al.1992). Thus, generation from the lexical level is not useful to many applications such as MT where the input to a realization component is usually a lexeme with a feature list. A solution to this problem was proposed by (Karttunen et al.1992) which involved composition of multiple FSTs that convert input from a deep level of representation to the lexical level. However, there are still many restriction on the order of elements presented as input and their compatibility.[1] The only work on Arabic morphological generation that focuses on generation issues is done by (Cavalli-Sforza et al.2000; Soudi et al.2001). Theirs is a lexeme-based approach that uses transformational rules to address the issue of stem change in various prefix/suffix contexts. Their system is a prototype that lacks in large scale coverage.

There are certain desiderata that should be expected from a morphological generation system for any language. These include (1) coverage of the language of interest in terms of both lexical coverage (large scale) and coverage of morphological and orthographic phenomena (robustness); (2) the surface forms are mapped from a deep level of representation that abstracts over language-specific morphological and orthographic features; and finally, (3) availability for the research community. These three issues were essential in the design of Aragen for Arabic morphological generation. Aragen[2] is a lexeme-based generation system that is built on top of a publicly available large-scale (albeit non-lexeme-based) database, Buckwalter's lexicon for morphological analysis. Lexemes, as opposed to stems, provide a desirable level of abstraction that is language independent for applications such as MT and NLG. Lexemes are also less abstract than roots, which tend to be semantically too ambiguous to be practically useful.

2 Buckwalter Morphological Analyzer

The Buckwalter morphological analyzer uses a concatenative lexicon-driven approach where morphotactics and orthographic rules are built directly into the lexicon itself instead of being specified in terms of general rules that interact to realize the output (Buckwalter2002). The system has three components: the lexicon, the compatibility tables and the analysis engine. An Arabic word is viewed as a concatenation of three regions, a prefix region, a stem region and a suffix region. The prefix and suffix regions can be null. Prefix and suffix lexicon entries cover all possible concatenations of Arabic prefixes and suffixes, respectively. For every lexicon entry, a morphological compatibility category, an English gloss and occasional part-of-speech (POS) data are specified. Stem lexicon entries are clustered around their specific lexeme, which is not used in the analysis process. See Figure 1.[3]

Compatibility tables specify which morphological categories are allowed to co-occur. For example, the morphological category for the prefix conjunction و/`w` (*and*), `Pref-Wa`, is compatible with all noun stem categories and perfect verb stem categories. However, `Pref-Wa` is not compatible with imperfective verb stems because they must contain a subject prefix. Similarly, the stem كتاب/`kitAb` of the the lexeme $_1$كتاب/`kitAb_1` (*book*) has the category (`Ndu`),

[1] Other work on using FSTs designed for analysis in generation is discussed in (Minnen et al.2000).

[2] The Aragen engine can be downloaded for free under an OpenSource license for research purposes from http://clipdemos.umiacs.umd.edu/Aragen. The lexical databases need to be acquired independently from the Linguistic Data Consortium (LDC) as part of the Buckwalter Arabic Morphological Analyzer (Buckwalter2002).

[3] All *romanized* Arabic examples are provided in the Buckwalter transliteration scheme (Buckwalter2002).

و/wa	Pref-Wa	*and*
ب/bi	NPref-Bi	*by/with*
وب/wabi	NPref-Bi	*and + by/with*
ال/Al	NPref-Al	*the*
بال/biAl	NPref-BiAl	*with/by + the*
وبال/wabiAl	NPref-BiAl	*and + with/by the*
ة/ap	NSuff-ap	*[fem.sg.]*
تان/atAni	NSuff-atAn	*two*
تين/atayoni	NSuff-tayn	*two*
تاه/atAhu	NSuff-atAh	*his/its two*
ات/At	NSuff-At	*[fem.pl.]*

;; $_1$كَتَب /katab-u$_1$		
كَتَب /katab	PV	*write*
كُتُب /kotub	IV	*write*
كُتِب /kutib	PV_Pass	*be written; be fated; be destined*
كُتَب /kotab	IV_Pass_yu	*be written; be fated; be destined*
;; $_1$كِتاب /kitAb$_1$		
كِتاب /kitAb	Ndu	*book*
كُتُب /kutub	N	*books*
;; $_1$كِتابة /kitAbap$_1$		
كِتاب /kitAb	Nap	*writing*

Figure 1: Some Buckwalter Lexical Entries

which is not compatible with the feminine marker ة/ap's category NSuff-ap. The same stem, كتاب/kitAb, appears as one of the stems of the lexeme $_1$كتابة/kitAbap$_1$ (*writing*) with a category that *requires* a suffix with the feminine marker. Cases such as these are quite common and pose a challenge to the use of stems as tokens since they add unnecessary ambiguity.

The analysis algorithm is rather simple since all of the hard decisions are coded in the lexicon and the compatibility tables: Arabic words are segmented into all possible sets of prefix, stem and suffix strings. In a valid segmentation, the three strings exist in the lexicon and are three-way compatible (prefix-stem, stem-suffix and prefix-suffix).

3 Aragen

The input to Aragen is a *feature-set*, a set of lexeme and features from a closed class of inflectional phenomena. These include number, gender and case inflections, which do appear in other languages, but also prefix conjunctions and prepositions that are written as part of the word in Arabic orthography. Aragen uses the Buckwalter lexicon described earlier *as is*. The lexicon is processed in Aragen to index entries based on inferred sets of features values (or *feature-keys*) that are used to map features in the input feature-sets to proper lexicon entries. This task is trivial for cases where the lexicon entry provides all necessary information. For example, verb voice and aspect are always part of the stem. For example, the feature-key for the stem of the passive perfective form of the verb كتب/katab is katab+PV+PASS.

Many lexicon entries, however, lack feature specifications. One example is broken plurals, which appear under their lexeme cluster, but are not marked in any way for plurality (see the entry for كتب/kutub in Figure 1). Detecting when a stem is plural is necessary to include the feature *plural* in the feature-key for that stem. Using the English gloss to detect the presence of a broken plural is a possible solution. However, it fails for adjectival entries since English adjectives do not inflect for plurality, e.g. كبير/kabiyr (SG) and كبار/kibAr (PL) are both glossed as *big*. Additionally, some sound plural stems in the lexicon are glossed as plurals. The Buckwalter categories are not helpful on their own for this task. For example, the presence of a stem with morphological category N is ambiguous as to being a broken plural or a singular nominalization of a form I verb (Buckwalter2002). The solution for this problem stems from the observation that a singular verbal nominalization is its own *lexeme*, whereas a broken plural is always listed under a lexeme that is in a singular base form. A broken plural is by definition a major change in the form of the lexeme. Therefore, if a stem under a lexeme has the morphological category N, Ndip, or Nap (all of which can mark a broken plural) AND it is **not** a subset string of the lexeme, it is considered a broken plural.

The process of generating from feature-sets is similar to Buckwalter analysis except that feature-

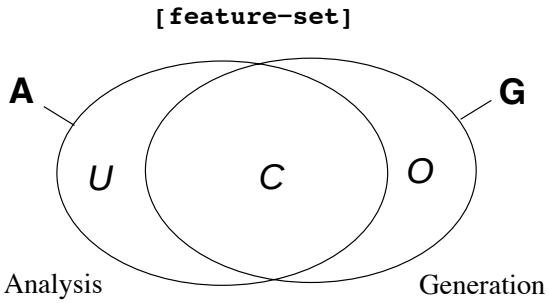

Figure 2: Aragen Evaluation

keys are used instead of string sequences. First, the feature-set is expanded to include all forms of underspecified obligatory features, such as case, gender, number, etc. Next, all feature-keys in the Aragen lexicon that fully match any subset of the expanded feature-set are selected. All combinations of feature-keys that completely cover the features in the expanded feature-set are matched up in prefix-stem-suffix triples. Then, each feature-key is converted to its corresponding prefix, stem or suffix. The same compatibility tables used in Buckwalter analysis are used to accept or reject prefix-stem-suffix triples. Finally, all unique accepted triples are concatenated and output. In the case that no surface form is found, a back-off solution that attempts to regenerate after discarding one of the input features is explored. If the back-off fails, typically due to a missing lexical entry, a baseline Arabic morphological generator is used.

The baseline generator uses a simple concatinative word structure rule and a small lexicon. The lexicon contains 70 entries that map all features to most common surface realizations. For example, `FEM` maps to (ة/`ap`, ت/`at`, and ϕ) and `PL` maps to (ات/`At`,ين/`iyna`,ي/`iy`,ون/`uwna` and و/`uw`). Subtleties of feature interaction are generally ignored except for the case of subject and verb aspect since the circumfix realization of subjects in the imperfective/imperative form is rather complex to model concatinatively. The only word structure rule used in the baseline generator is the following:

```
<WORD> ::= (w|f) (s|l|b|k) Al <SubjectAspect>
           <Lexeme>
           <AspectSubject> <Gender> <Number> <Object> <Possessive>
```

4 Evaluation

A sample text of over one million Arabic words from the UN Arabic-English corpus (Jinxi2002) was used in this evaluation. For each unique word in the text, the Buckwalter morphological analyzer is run with a simple rewriting of its output to match the format of Aragen feature-set input. The resulting feature-sets are then input to two systems: the complete Aragen as described earlier *and* the baseline generator used as back-off to Aragen . For each feature-set, there are two sets of words: (a) words that analyze into the feature-set (A words) and (b) words that are generated from the feature-set (G words) (see Figure 2). The bigger the intersection between the two sets (C words), the better the performance of a system. Generated words that are not part of the intersection (C words) are Overgenerated words (O words). Words that analyze into the feature-set but are not generated are Undergenerated words (U words). In principle, U words are definite signs of problems in the generation system; whereas, O words can be correct but unseen in the analyzed text.

Table 1: Evaluation Results

System	UnderErr	OverErr	CombErr	Time (CPU secs)
Aragen *diacritized*	0.39%	12.22%	0.76%	1,769
Aragen *undiacritized*	0.38%	12.42%	0.74%	1,745
Baseline *diacritized*	43.90%	60.99%	51.05%	281
Baseline *undiacritized*	32.84%	47.93 %	38.98%	293

A system's Undergeneration Error (UnderErr) is defined as the ratio of U words to A words. Overgeneration Error (OverErr) is defined as the ratio of O words to G words. These two measure are quite similar to (1 - Precision) and (1- Recall) respectively, if the set of A words paired with a feature-set are considered a gold standard to be replicated in reverse by a generation system. The Combined Undergeneration and Overgeneration Error (CombErr) is calculated as their harmonic mean (in a manner similar to calculating the F-scores for Precision and Recall)

$$\text{UnderErr} = \frac{U}{A} = \frac{A-C}{A} \qquad \text{OverErr} = \frac{O}{G} = \frac{G-C}{G} \qquad \text{CombErr} = \frac{2 \times UnderErr \times OverErr}{UnderErr + OverErr}$$

The evaluation text contained 63,066 undiacritized unique words, which were analyzed into 118,835 unique feature-sets corresponding to 14,883 unique lexemes. The number of unique diacritized words corresponding to the text words is 104,117. The evaluation was run in two modes controlling for the type of matching between A words and G words: diacritized (or diacritization-sensitive) and undiacritized. Evaluation results comparing Aragen to baseline are presented in Table 1. The baseline system is almost six times faster than Aragen[4], but it had a high undergeneration and overgeneration error rates. Both were reduced in the undiacritized mode, where some erroneous output became ambiguous with correct output. Aragen, by comparison, reduced the error rate from the baseline by more than one order of magnitude overall.

Many of the overgeneration errors are false alarms. They include cases of overgeneration of broken plurals, some of which archaic or genre-specific but correct. For example, the word for *Sheik*, شيخ/`$yx`, has three uncommon broken plurals in addition to the common شيوخ/*$ywx*: اشياخ/*>a$oyAx*, مشايخ/*ma$Ayix*, and مشائخ/*ma$A}ix*. Another very common overgeneration error resulted from the underspecification of some mood-specific vocalic verbal suffixes in the Buckwalter lexicon. Arabic hollow verbs, for example, undergo a stem change in the jussive mood (from *yaquwl* to *yaqul*), which is indistinguishable in the analysis.

Undergeneration errors stem exclusively from lexicon errors. These are not many and they can be expected in a manually created database. One example is caused by a missing lexeme comment in the Buckwalter lexicon which resulted in pairing all the forms of the verb رأى/`r>Y` (*to see*) to the lexeme that appears just before it, راوند/`rAwanod` (*rhubarb*). Such cases suggest a valuable use of Aragen as a debugging tool for the Buckwalter lexicon.

5 Conclusions and Future Work

Aragen is a large-scale lexeme-based Arabic morphological generation system that retargets the databases of the Buckwalter Arabic analyzer for generation purposes. An evaluation of Aragen in terms of Overgeneration/ Undergeneration Error Rates indicates it to be more than one order of magnitude better than a simple baseline. Future work includes evaluating Aragen in the context of an MT or NLG system and extending Aragen to handle Arabic dialect morphology.

[4]The experiments were run on a Dell Inspiron machine with Pentium 4 CPU and 2.66 GHz.

Acknowledgments

This work has been supported, in part, by the National Science Foundation grant 0329163, "Arabic Dialect Modeling for Speech and Natural Language Processing," and by Army Research Lab Cooperative Agreement DAAD190320020. I would like to thank Owen Rambow and Mona Diab for helpful discussions.

Références

M. Aljlayl and O. Frieder. 2002. On arabic search: Improving the retrieval effectiveness via a light stemming approach. In *Proceedings ofACM Eleventh Conference on Information and Knowledge Management, Mclean, VA*.

K. Beesley. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages volume 1, 89–94, Copenhagen, Denmark.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.

Violetta Cavalli-Sforza, Abdelhadi Soudi, and Teruko Mitamura. 2000. Arabic Morphology Generation Using a Concatenative Strategy. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, pages 86–93, Seattle, Washington, USA.

Kareem Darwish. 2002. Building a Shallow Morphological Analyzer in One Day. In *Proceedings of the workshop on Computational Approaches to Semitic Languages in the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA, USA.

Xu Jinxi. 2002. UN Parallel Text (Arabic-English), LDC Catalog No.: LDC2002E15. Linguistic Data Consortium, University of Pennsylvania.

L. Karttunen, R. Kaplan, and A. Zaenen. 1992. Two-level morphology with composition. In *Proceedings of Fourteenth International Conference on Computational Linguistics (COLING-92)*, pages 141–148, Nantes, France, July 20–28.

George Kiraz. 1994. Multi-tape Two-level Morphology: A Case study in Semitic Non-Linear Morphology. In *Proceedings of Fifteenth International Conference on Computational Linguistics (COLING-94)*, pages 180–186, Kyoto, Japan.

K. Koskenniemi. 1983. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685.

Guido Minnen, John Carroll, and Darren Pearce. 2000. Robust, Applied Morphological Generation. In *Proceedings of the 1st International Conference on Natural Language Generation (INLG 2000)*, Mitzpe Ramon, Israel.

A. Soudi, V. Cavalli-Sforza, and A. Jamari. 2001. A Computational Lexeme-Based Treatment of Arabic Morphology. In *Proceedings of the Arabic Natural Language Processing Workshop, Conference of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France.