Feature-Based Tagger of Approximations of Functional Arabic Morphology

Jan Hajič & Otakar Smrž
Inst. of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University in Prague
{hajic,smrz}@ufal.mff.cuni.cz

Tim Buckwalter & Hubert Jin
Linguistic Data Consortium
University of Pennsylvania
timbuck2@ldc.upenn.edu
hubertj@ldc.upenn.edu

1 Introduction

The field of morphological disambiguation of Arabic has recently witnessed significant achievements (Habash and Rambow [15], Smith et al. [28]). Through them, the Penn Arabic Treebank (PATB, Maamouri et al. [24]) is being confirmed as a standard for development and evaluation of systems for automatic morphological processing of Arabic, and the Buckwalter Arabic Morphological Analyzer (Buckwalter [6, 7]) is becoming the most respected lexical resource of its kind.

The context for understanding the current paper has evolved since our work on it started, yet, the motivation for it is unchanged and the conclusions are valid and up-to-date. We would like to open some issues concerning the very description of Arabic morphology and point out that in this domain, one should carefully distinguish individual problems, theories, resources, and solutions for their frequent idiosyncrasies and incompatibilities.

In this contribution, we reference Functional Arabic Morphology (Smrž [29]) and take the Buckwalter Morphology as the departure point for approximating this novel model by (a) restoring the true syntactic units (b) seeking their functional, rather than structural, morphological categories. We then present five versions of a feature-based morphological tagger depending on that approximation, which were built on all the currently available Parts of PATB, as well as on the MorphoTrees annotations of the Prague Arabic Dependency Treebank (PADT, Hajič et al. [18]).

1.1 The Disambiguation Problem

Arabic is a language of rich morphology, both derivational and inflectional (Holes [20]). Due to the fact that the Arabic script usually does not encode short vowels and omits some other important phonological distinctions, the degree of morphological ambiguity is very high.

In addition, Arabic orthography prescribes to concatenate certain word forms with the preceding or the following ones, which makes the boundaries of syntactic units, i.e. **tokens** as we denote them, obscure. Unlike in Chinese or German, however, in Arabic there are clear limits to the number and the kind of tokens that can combine in this manner. What appears to be one orthographical **string** in a Modern Standard Arabic text, can actually constitute from up to four syntactic tokens.[1]

In Latin script-based languages, one usually assumes that words, i.e. the input strings, can be processed into tokens easily and *uniquely* by an independent tokenizer module that runs *before* a morphological tagger. For Arabic, this is not possible — one input string can be analyzed in such ways that not only the morphemes, but even the syntactic tokens may vary for individual readings of the string.

Thus, the problem of disambiguation of this language encompasses subproblems like tokenization, full morphological tagging or its simplified 'part-of-speech' versions, lemmatization, diacritization (discussed in Nelken and Shieber [25]) or restoration of the structural components of words. These subproblems, of course, can come in many variants and combinations.

1.2 Existing Morphological Systems

The long evolution of computational modeling of Arabic morphology is nowadays mirrored in the excellent works of (Kiraz [23]) and (Beesley and Karttunen [4]), and although many morphological systems are in development (Ramsay and Mansur [27] or Soudi et al. [32], inter alia), only (Beesley [3]) and (Buckwalter [6, 7]) are actually accessible to the interested public, meeting the prerequisite to their wider application and evaluation.

It appears from the literature and implementations (many summarized in Al-Sughaiyer and Al-Kharashi [1]) that Arabic computational morphology has understood its purpose in the sense of operations with morphs rather than morphemes (cf. El-Sadany and Hashish [12]; see also Sproat [33] or Stump [34]), and has not concerned itself systematically and to the necessary extent with its role for syntax.

The outline of formal grammar in (Ditters [11]), for instance, works with grammatical categories like number, gender, humanness, definiteness, but one cannot see which of the existing systems could provide for this information correctly, as they *misinterpret* some morphs for bearing a category, and *underdetermine* lexical morphemes in general. Certain syntactic parsers, like (Othman et al. [26]), may resort to their own morphological analyzers, but still, they do not get rid of the

[1]In theory (Fischer [13]), an additional personal suffix might increase this number to five, which is extremely unlikely to occur in the standard language, and is unattested in the resources we study.

String ⋯ Token	Token Tag	Buckwalter's Morph Tags	Token Form	Token Gloss
سيخبرهم	`F---------`	FUT	*sa-*	will
	`VIIA-3MS--`	IV3MS+IV+IVSUFF_MOOD:I	*yu-ḫbir-u*	he-notify
	`S----3MP4-`	IVSUFF_DO:3MS	*-hum*	them
بذلك	`P---------`	PREP	*bi-*	about/by
	`SD----MS--`	DEM_PRON_MS	*ḏālika*	that
عن	`P---------`	PREP	*ʿan*	by/about
طريق	`N-------2R`	NOUN+CASE_DEF_GEN	*ṭarīq-i*	way-of
الرسائل	`N-------2D`	DET+NOUN+CASE_DEF_GEN	*ar-rasāʾil-i*	the-messages
القصيرة	`A-----FS2D`	DET+ADJ+NSUFF_FEM_SG+ +CASE_DEF_GEN	*al-qaṣīr-at-i*	the-short
والإنترنت	`C---------`	CONJ	*wa-*	and
	`Z-------2D`	DET+NOUN_PROP+ +CASE_DEF_GEN	*al-ʾinternet-i*	the-internet
وغيرها	`C---------`	CONJ	*wa-*	and
	`FN------2R`	NEG_PART+CASE_DEF_GEN	*ġayr-i*	other/not-of
	`S----3FS2-`	POSS_PRON_3FS	*-hā*	them

Figure 1: Tokenization of input strings into tokens in *he will notify them about that through SMS messages, the Internet, and other means*, and the disambiguated morphological analyses with Buckwalter's tags and the quasi-functional token tags.

form of an expression and only incidentally introduce truly functional categories (cf. Hajič et al. [19]). In syntactic considerations they often call for discriminative semantic features instead. Commercial systems, esp. (Chalabi [9]), do not seem to overcome this interference either.

The missing common rationale as to what higher linguistic framework the morphology should serve for crystalizes in the number of individual, ad hoc tagsets and a very rare discussion of their motivation, completeness, relevance and actual expressive power. This situation brought us to designing Functional Arabic Morphology.[2] In (Hajič et al. [19], Smrž and Hajič [30]), we discuss its principles and show how e.g. agreement can naturally be controlled and restored in the functional system — which is impossible if recognizing morphs only and not their functions.

1.3 Approximating the Functional Model

The underlying morphological engine for both the Penn Arabic Treebank and the Prague Arabic Dependency Treebank is the Buckwalter Arabic Morphological An-

[2]Functional Arabic Morphology (Smrž [29]) is being implemented in the Functional Morphology (Forsberg and Ranta [14]), which is a methodology as well as a domain-specific programming language embedded in Haskell building on the computational toolkit Zen for Sanskrit (Huet [21, 22]).

Morphs	Form	Token Tag	Lemma	Glosses per Morph
`|laY+(null)`	*ʾālā*	`VP-A-3MS--`	*ʾālā*	promise/take an oath + he/it
`|liy˜`	*ʾālīy*	`A---------`	*ʾālīy*	mechanical/automatic
`|liy˜+u`	*ʾālīy-u*	`A-------1R`	*ʾālīy*	mechanical . . . + [def.nom.]
`|liy˜+i`	*ʾālīy-i*	`A-------2R`	*ʾālīy*	mechanical . . . + [def.gen.]
`|liy˜+a`	*ʾālīy-a*	`A-------4R`	*ʾālīy*	mechanical . . . + [def.acc.]
`|liy˜+N`	*ʾālīy-un*	`A-------1I`	*ʾālīy*	mechanical . . . + [indef.nom.]
`|liy˜+K`	*ʾālīy-in*	`A-------2I`	*ʾālīy*	mechanical . . . + [indef.gen.]
`|l` +	*ʾāl*	`N--------R`	*ʾāl*	family/clan +
+ `iy`	*-ī*	`S----1-S2-`	*ī*	+ my
`IilaY`	*ʾilā*	`P---------`	*ʾilā*	to/towards
`Iilay` +	*ʾilay*	`P---------`	*ʾilā*	to/towards +
+ `ya`	*-ya*	`S----1-S2-`	*ya*	+ me
`Oa+liy+(null)`	*ʾa-lī*	`VIIA-1-S--`	*waliya*	I + follow/come after + [ind.]
`Oa+liy+a`	*ʾa-liy-a*	`VISA-1-S--`	*waliya*	I + follow/come after + [sub.]

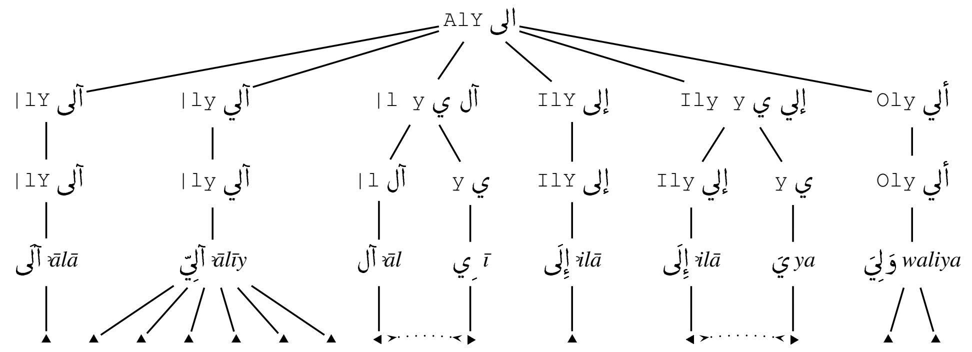

Figure 2: Analyses of the string `AlY` الى turned into the MorphoTrees hierarchy.

alyzer (Buckwalter [6, 7]). While PATB adopts the analyses in their original format (Maamouri et al. [24]), the PADT annotations take place on *quasi-functional* approximations organized into MorphoTrees (Smrž and Pajas [31]).

With respect to the linguistic view and the architecture of the tagger that we will develop, we unify the format of the morphological data by converting all the Parts of PATB into the approximation, which is done in two steps: (a) the morphs of the original input strings are re-grouped to form tokens (b) the corresponding sequences of tags are mapped into the fixed-width positional notation of PADT.

Let us illustrate the transformations through Figure 1 and Figure 2, and refer to (Smrž and Pajas [31], Smrž and Hajič [30]) for any other details.

2 The Feature-Based Tagger

The refined morphological information that we have sought in our studies requires full morphological tagging to be established. The Arabic tagger that we present is an adaptation of the feature-based, exponential-model tagger described in (Hajič and Hladká [17]), taking the advantage of the positional tag system by predicting the individual columns/categories separately (but not *irrespective* of the other ones in context). It has been used on several inflectional and agglutinative languages.

Recall the problem of tokenization being part of morphological analysis in Arabic. In order to keep the tagger's functionality unmodified, we extend the tagset in the way that all input strings will be considered 4-tuples of tokens, with the resulting **string tag** being a concatenation of the **token tags**. By disambiguating such aggregate tags, the tagger decides the tokenization as well, since the tokens can be deterministically derived from the tag and the list of associated lemmas.[3]

2.1 Feature-Based Tagging

Instead of employing the source–channel paradigm for tagging, we are using here a conditional approach to modeling, for which we have chosen an exponential probabilistic model. Such model (when predicting an event[4] $y \in Y$ in a context x) has the general form

$$p_{AC,e}(y|x) = \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(y,x))}{Z(x)} \tag{1}$$

where $f_i(y,x)$ is the set (of size n) of binary-valued (yes/no) *features* of the event value being predicted and its context, λ_i is a "weight" (in the exponential sense) of the feature f_i, and the normalization factor $Z(x)$ is defined naturally as

$$Z(x) = \sum_{y \in Y} \exp(\sum_{i=1}^{n} \lambda_i f_i(y,x)) \tag{2}$$

We use a separate model for each ambiguity class AC (that actually appeared in the training data) of each of the 4×10 morphological categories. The final distribution $p_{AC}(y|x)$ is further smoothed using unigram distributions on subtags (again, separately for each category):

$$p_{AC}(y|x) = \sigma p_{AC,e}(y|x) + (1-\sigma) p_{AC,1}(y) \tag{3}$$

Such smoothing takes care of any unseen context; for ambiguity classes not seen in the training data, for which there is no model, we use unigram probabilities of subtags, one distribution per category.

[3] Lemma disambiguation is a separate process following tagging, and is not covered here.

[4] In our case, a **subtag**, i.e. a unique value of a morphological category.

In the general case, features can operate on any imaginable context. In practice, we view the context as a set of attribute–value pairs with a discrete range of values. Every feature can thus be represented by a set of contexts in which it is positive. There is, of course, also a distinguished attribute for the value of the variable being predicted (y); the rest of the attributes is denoted by x as expected. Values of attributes are denoted by an overstrike ($\overline{y}, \overline{x}$).

The pool of contexts of prospective features is for the purpose of morphological tagging defined as a full cross-product of the category being predicted (y) and of the x specified as a combination of

> [A] an ambiguity class of a single category, which may be different from the category being predicted, or [B] a word form (the input string), or [C] a single position value membership in an ambiguity class, or [D] a full tag (to the left of the current position only),

and

> [E] the current position, or [F] immediately preceding/following position in text, or [G] position $\pm$ 2 strings apart, or [H] closest preceding/following position (up to four positions away) having a certain ambiguity class in the POS category.

The full cross-product of these contexts is prohibitively large, but there are means to limit the size of the pool of features to fit to available memory. For Arabic, we have used a limit of 7 million feature contexts in the pool.

Feature weights can be computed only iteratively, but it is impossible to do so in reasonable time while selecting the features at the same time, even when using certain shortcuts (Berger et al. [5]). Therefore, the initial feature weight of a feature which is true in context $\overline{x}$ for a tag $\overline{y}$ is estimated as the log of the conditional probability $p(\overline{y}|\overline{x})$, estimated by MLE from the training data. This makes the model essentially a form of a Naive Bayes one.

The learner is allowed to vary the weights (in several discrete steps) during feature selection, a (somewhat crude) attempt to depart from the original Naive Bayes simplification to the approximation of the "correct" Maximum Entropy estimation.

2.2 Training Iterations

Given the huge number of possible features, the training proceeds in four "iterations", each adding more complex features, but only from those training events that are in error after the previous iteration.

The training "iterations" are not really part of the algorithm; they only allow to try and possibly keep more detailed features in later iterations when most simple

Characteristics	# Train	# Test Data		# String Tags		# Token Tags	
Experiment	Strings	Strings	Tokens	Total	Anno.	Total	Anno.
PATB Part 2 Prototype	122 556	19 683	23 074	2 031	852	317	242
PATB Part 3	320 998	19 283	22 690	2 864	1 251	391	314
PADT MorphoTrees	106 887	19 253	22 547	3 164	927	378	265
PATB Part 1	120 045	19 339	22 131	884	534	165	143
PATB Part 1 Revised	125 392	19 363	22 104	2 226	785	401	271

Table 1: Characteristics of the sets of data (note all tags vs. annotated only).

features are not generated any more since there are much fewer errors remaining. Experimentally, we found that adding more than four such iterations does not improve the results. The type and nature of the features allowed for the i-th iteration has been tried experimentally and heuristically. Typically, only the simplest features (such as those having an input string as the only "context" and nothing more) are used in the first iteration.

A single training iteration first generates a feature pool of a predetermined size and of the requested type (and complexity), and then proceeds in selecting features (and estimating their weights) in a greedy way (minimizing the training data error rate directly as the objective function) as described in (Hajič [16]).

3 Experiments and Evaluation

Table 1 overviews the five experiments described below, the parameters of which severely differ. The level of detail of morphological annotation (cf. sizes of tagsets) and the ambiguity within analyses considerably increase with the progress in time.

The resulting models were tested using the simplest tagging mode (Viterbi beam width 1, effectively canceling the Viterbi search,[5] and the independence assumption about the categories, i.e., simply multiplying the probabilities of the 40 category values and normalizing by the available tags listed as analyses).

Table 2 delivers the taggers' performance in terms of accuracy for full morphological tagging (40 positions per string, 10 positions per token), part-of-speech tagging (assigning only the first position, one of 15 values, in each token tag), and lemmatization (choosing one lemma for every token in the string). Two variants of tokenization are evaluated using $F_{\beta=1}$ (see further the Discussion).

[5]Please note that this is not a decisive factor for the resulting accuracy, as opposed to e.g. HMM--based tagging, since the tagger looks right for morphological ambiguity classes — see above the description of the feature pool available to the tagger at training time.

Performance	Per String	Per Token				
Experiment	Full (40)	Full (10)	**POS**	Lemma	Tknz++	**Tknz**
PATB Part 2 Prototype	87.88	89.31	96.46	92.33	99.31	99.51
PATB Part 3	86.82	88.17	95.25	89.91	97.52	98.60
PADT MorphoTrees	87.73	89.24	96.02	90.64	97.71	99.25
PATB Part 1	96.85	96.99	97.37	92.75	97.47	99.37
PATB Part 1 Revised	88.13	89.16	95.57	90.27	97.13	98.86

Table 2: Performance evaluation for the individual experiments (in percents).

Penn Arabic Treebank Part 2, Version 1 The pre-release of this dataset (identified as LDC2003E17) served for developing the prototypes of both the tagger and the mapping between Buckwalter's sequences of tags and the quasi-functional positional token tags. The analyses do not seem to overgenerate for orthographical variation (Buckwalter [8]) too much yet (note the similar Tknz++ and Tknz).

Habash and Rambow [15] report 96.5 % accuracy in POS tagging for the comparable dataset and tagset, counting, unlike us, only the well-tokenized data.

Penn Arabic Treebank Part 3, Version 1 This dataset (LDC2004T11) brings the advanced features of Buckwalter's morphology, among which are complete vocalization (with case and mood endings), extended lexicon, and finer tags for verbs and particles. Therefore, the mapping into the approximation also improved, and the complexity of the tagset largely increased compared to that of the prototype.

Prague Arabic Dependency Treebank 1.0 Due to the nature of MorphoTrees (LDC2004T23), where long-dependency relations between tokens may be weakened and some values in tags expanded for the sake of more precise annotations, certain token combinations may be listed in the format for the tagger that the analyzer would not produce (note the highest number of non-annotated string tags).

MorphoTrees are the 'purest' available approximation of the Functional Arabic Morphology. Given the detail and the complexity of the data, the tagger's performance is remarkable. Just like with PATB Part 3, no other computational results relevant to this dataset are known to us.

Penn Arabic Treebank Part 1, Version 2 The annotations in this dataset (LDC-2003T06) are morphologically most 'impoverished', but have been used by other researchers (Diab et al. [10], Habash and Rambow [15]) to train very successful taggers based on support vector machines. Habash and Rambow [15] reach 98.1 % of POS accuracy and 96.2 % of accuracy in full token tagging, which are results well comparable to ours.

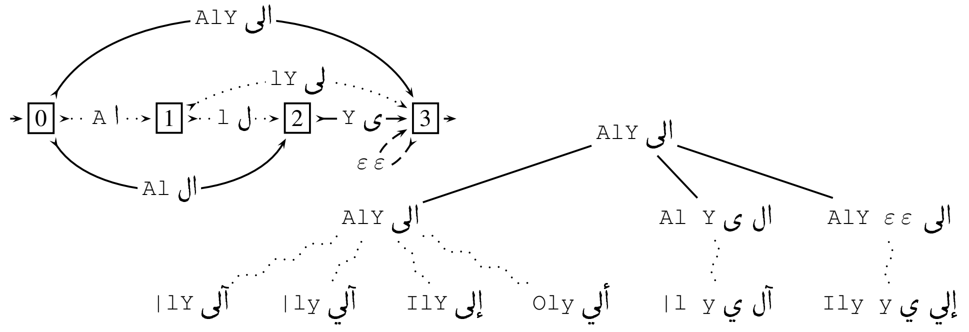

Figure 3: Discussion of partitioning and tokenization of input strings.

Penn Arabic Treebank Part 1, Version 3 This is the revised version (LDC2005-T02) of the previously mentioned corpus, with a complete coverage of the lexicon and including the advanced features of the morphology.

It is actually this dataset and not its former version (Smith et al., p.c.) that Smith et al. [28] used for the development of their log-linear source–channel tagging model. It works with strings and morphs only, but achieves overwhelming results (accuracy) — 96.1 % in full string tag disambiguation, 95.4 % in the restoration of morphs, and 94.6 % in assigning one representative lemma per input string.

4 Discussion and Conclusions

We have introduced two measures for tokenization. Tknz is close to the evaluations in (Habash and Rambow [15], Diab et al. [10]) which only check the partitioning determined by finding token boundaries between the characters of the original string, and do not, unlike Tknz++, require the tokenization to faithfully reconstruct the canonical non-vocalized forms of tokens, as is the standard in MorphoTrees (Smrž and Pajas [31], Smrž and Hajič [30]).

The disparity of these tokenizations is illustrated in Figure 3. The graph on the left depicts the three 'sensible' ways of partitioning the input string AlY الى in the approach of (Diab et al. [10]), where characters are classified to be token-initial or not. Two tokenizations are obtained by linking the boundaries from 0 to 3 following the solid edges. The third partitioning AlY $\varepsilon\,\varepsilon$ الى implies there is another fictitious boundary and some 'empty word' $\varepsilon\,\varepsilon$ at the end of the string, which corresponds to taking the dashed edge in the graph.

Even though conceptually sound, this kind of partitioning cannot undo the effects of orthographical variation (Buckwalter [8]), nor express other useful distinctions. The hierarchy in Figure 3 relates this tokenization to that of Figure 2. Habash and Rambow [15, section 7] correctly point out that "[t]here is not a single possible or obvious tokenization scheme: a tokenization scheme is an analytical tool devised by the researcher." Nonetheless, different tokenizations capture different information, and some may be linguistically not as appropriate as others (cf. Bar-Haim et al. [2] for the influence of tokenization on tagging in Hebrew).

In any case, we evaluate tokenizations in terms of the Longest Common Subsequence (LCS) problem. The tokens that are the members of the LCS with some referential tokenization, are considered correctly recognized. Dividing the length of the LCS by the length of one of the sequences, we get recall, doing it for the other of the sequences, we get precision. The harmonic mean of both is $F_{\beta=1}$.

We have presented five versions of the feature-based tagger of Arabic, developed gradually on all the data of the Penn Arabic Treebank and the Prague Arabic Dependency Treebank. Using the experience with other inflectional languages, we prefer the functional treatment of the morphology of Arabic, which we now only approximate. The pure description with respect to syntactic tokens and their relevant, functional grammatical categories is being further pursued and implemented.

The results of our tagger rank competitively high in the field (cf. Habash and Rambow [15]). Full morphological tagging is expected to improve with the increasing 'functionality' of the data. Note that applying the conditionally-estimated context-based models set forth in (Smith et al. [28]) to such data is certainly possible and promising, too. Lemmatization and the issue of unknown words have only received little attention in our tagger, and can be well improved.

This research was supported by the Ministry of Education of the Czech Republic, project MSM0021620838, by the Grant Agency of Charles University in Prague, project 207-10/203333, and through the Fulbright-Masaryk Fellowship of the Fulbright Commission in the Czech Republic.

References

[1] Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi. Arabic Morphological Analysis Techniques: A Comprehensive Survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213, 2004.

[2] Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. Choosing an Optimal Architecture for Segmentation and POS-Tagging of Modern Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 39–46, Ann Arbor, 2005.

[3] Kenneth R. Beesley. Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 1–8, Toulouse, France, 2001.

[4] Kenneth R. Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, California, 2003.

[5] Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, 1996.

[6] Tim Buckwalter. Buckwalter Arabic Morphological Analyzer Version 1.0. LDC catalog number LDC2002L49, ISBN 1-58563-257-0, 2002.

[7] Tim Buckwalter. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0, 2004.

[8] Tim Buckwalter. Issues in Arabic Orthography and Morphology Analysis. In *Proceedings of the COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages*, pages 31–34, 2004.

[9] Achraf Chalabi. Sakhr Arabic Lexicon. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 21–24. ELDA, 2004.

[10] Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *HLT-NAACL 2004: Short Papers*, pages 149–152, 2004.

[11] Everhard Ditters. A Formal Grammar for the Description of Sentence Structure in Modern Standard Arabic. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 31–37, Toulouse, France, 2001.

[12] Tarek A. El-Sadany and Mohamed A. Hashish. An Arabic morphological system. *IBM Systems Journal*, 28(4):600–612, 1989.

[13] Wolfdietrich Fischer. *A Grammar of Classical Arabic*. Yale Language Series. Yale University Press, third revised edition, 2001. Translated by Jonathan Rodgers.

[14] Markus Forsberg and Aarne Ranta. Functional Morphology. In *Proceedings of ICFP 2004*, pages 213–223. ACM Press, 2004.

[15] Nizar Habash and Owen Rambow. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics ACL 2005*, pages 573–580, Ann Arbor, 2005.

[16] Jan Hajič. Morphological Tagging: Data vs. Dictionaries. In *Proceedings of NAACL-ANLP 2000*, pages 94–101, Seattle, 2000. ACL.

[17] Jan Hajič and Barbora Hladká. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of COLING-ACL 1998*, pages 483–490, Montreal, Canada, 1998. ACL.

[18] Jan Hajič, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kráčmar, and Kamila Hassanová. Prague Arabic Dependency Treebank 1.0. LDC catalog number LDC2004T23, ISBN 1-58563-319-4, 2004.

[19] Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, and Emanuel Beška. Prague Arabic Dependency Treebank: Development in Data and Tools. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 110–117. ELDA, 2004.

[20] Clive Holes. *Modern Arabic: Structures, Functions, and Varieties*. Georgetown Classics in Arabic Language and Linguistics. Georgetown University Press, 2004.

[21] Gérard Huet. The Zen Computational Linguistics Toolkit. ESSLLI Course Notes, FoLLI, the Association of Logic, Language and Information, 2002.

[22] Gérard Huet. A Functional Toolkit for Morphological and Phonological Processing, Application to a Sanskrit Tagger. *Journal of Functional Programming*, 2004.

[23] George Anton Kiraz. *Computational Nonlinear Morphology with Emphasis on Semitic Languages*. Studies in Natural Language Processing. Cambridge University Press, 2001.

[24] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 102–109. ELDA, 2004.

[25] Rani Nelken and Stuart M. Shieber. Arabic Diacritization Using Finite-State Transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86, Ann Arbor, 2005.

[26] Eman Othman, Khaled Shaalan, and Ahmed Rafea. A Chart Parser for Analyzing Modern Standard Arabic Sentence. In *Proceedings of the MT Summit IX Workshop on Machine Translation for Semitic Languages: Issues and Approaches*, pages 37–44, 2003.

[27] Allan Ramsay and Hanady Mansur. Arabic morphology: a categorial approach. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 17–22, Toulouse, France, 2001.

[28] Noah A. Smith, David A. Smith, and Roy W. Tromble. Context-Based Morphological Disambiguation with Random Fields. In *Proceedings of HLT/EMNLP 2005*, Vancouver, 2005.

[29] Otakar Smrž. *Functional Arabic Morphology. Formal System and Implementation*. PhD thesis, Charles University in Prague, in prep.

[30] Otakar Smrž and Jan Hajič. The Other Arabic Treebank: Prague Dependencies and Functions. In *Arabic Computational Linguistics: Current Implementations*. CSLI Publications, to appear.

[31] Otakar Smrž and Petr Pajas. MorphoTrees of Arabic and Their Annotation in the TrEd Environment. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 38–41. ELDA, 2004.

[32] Abdelhadi Soudi, Violetta Cavalli-Sforza, and Abderrahim Jamari. A Computational Lexeme-Based Treatment of Arabic Morphology. In *EACL 2001 Workshop Proceedings on Arabic Language Processing: Status and Prospects*, pages 155–162, Toulouse, 2001.

[33] Richard Sproat. *Morphology and Computation*. ACL–MIT Press Series in Natural Language Processing. MIT Press, 1992.

[34] Gregory T. Stump. *Inflectional Morphology. A Theory of Paradigm Structure*. Cambridge Studies in Linguistics. Cambridge University Press, 2001.