Automatic Morphological Enrichment of a Morphologically Underspecified Treebank

Sarah Alkuhlani, Nizar Habash and Ryan Roth
Center for Computational Learning Systems
Columbia University
{salkuhlani,habash,ryanr}@ccls.columbia.edu

Abstract

In this paper, we study the problem of automatic enrichment of a morphologically underspecified treebank for Arabic, a morphologically rich language. We show that we can map from a tagset of size six to one with 485 tags at an accuracy rate of 94%-95%. We can also identify the unspecified lemmas in the treebank with an accuracy over 97%. Furthermore, we demonstrate that using our automatic annotations improves the performance of a state-of-the-art Arabic morphological tagger. Our approach combines a variety of techniques from corpus-based statistical models to linguistic rules that target specific phenomena. These results suggest that the cost of treebanking can be reduced by designing underspecified treebanks that can be subsequently enriched automatically.

1 Introduction

Collections of manually-annotated morphological and syntactic analyses of sentences, or treebanks, are an important resource for building statistical parsing models or for syntax-aware approaches to applications such as machine translation. Rich treebank annotations have also been used for a variety of natural language processing (NLP) applications such as tokenization, diacritization, part-of-speech (POS) tagging, morphological disambiguation, base phrase chunking, and semantic role labeling.

The development of a treebank with rich annotations is demanding in time and money, especially for morphologically complex languages. Consequently, the richer the annotation, the slower the annotation process and the smaller the size of the treebank. As such, a tradeoff is usually made between the size of the treebank and the richness of its annotations.

In this paper, we investigate the possibility of automatically enriching the morphologically underspecified Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009; Habash et al., 2009) with the more complex POS tags and lemmas used in the Penn Arabic Treebank (PATB) (Maamouri et al., 2004). We employ a variety of techniques that range from corpus-based statistical models to handwritten rules based on linguistic observations. Our best method reaches accuracy rates of 94%-95% on full POS tag identification. We can also identify the unspecified lemmas in CATiB with an accuracy over 97%. 37% of our POS tag errors are due to gold tree or gold POS errors. A learning curve experiment to evaluate the dependence of our method on annotated data shows that while the quality of some components may reduce sharply with less data (12% absolute reduction in accuracy when using $\frac{1}{32}$ of the data or some 10K annotated words), the overall effect is a lot smaller (2% absolute drop). These results suggest that the cost of treebanking can be reduced by designing underspecified treebanks that can be subsequently enriched automatically.

The rest of this paper is structured as follows: Section 2 presents related work; Section 3 details various language background facts about Arabic and its treebanking; Section 4 explains our approach; and Section 5 presents and discusses our results.

2 Related Work

Arabic Treebanking There has been a lot work on building treebanks for different languages. In the case of Modern Standard Arabic (MSA), there are three efforts that vary in terms of richness and representation choice. The Penn Arabic Treebank (PATB) (Maamouri et al., 2004; Maamouri et al., 2009b; Maamouri et al., 2009a), the Prague Arabic Dependency Treebank (PADT) (Smrž and Hajič, 2006; Smrž et al., 2008) and the Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009; Habash et al., 2009) . The PATB uses phrase structure representation, while the other two use two

Proceedings of NAACL-HLT 2013, pages 460–470,
Atlanta, Georgia, 9–14 June 2013.

different dependency representations. The PATB and PADT representations are quite detailed. The PATB not only provides tokenization, complex POS tags (485 tags in our data set), and syntactic structure; it also provides empty categories, diacritization, lemma choices, glosses and some semantic tags. In comparison CATiB only provides tokenization, six POS tags and eight dependency relations. The tradeoff is speed: CATiB's complete POS and syntax annotation rate is 540 tokens/hour (and annotator training takes two months), a much higher speed than reported for complete (POS and syntax) annotation in PATB (around 250-300 tokens/hour and 6-12 months for annotator training) and PADT (around 75 tokens/hour) (Habash and Roth, 2009). An important recent addition to the family of Arabic treebanks is the Quran Treebank, which targets the Classical Arabic language of the Quran, not MSA (Dukes and Buckwalter, 2010).

Treebank Enrichment There has been a number of efforts on developing treebanks with rich representations and on treebank enrichment for many languages, such Danish, English, German, Italian and Spanish (Oepen et al., 2002; Hinrichs et al., 2004; Müller, 2010). Additionally, there has been some work on Arabic treebank enrichment that built on the PATB by manually extending its already rich annotations or automatically converting them to new formalisms. The Arabic Propbank (Propositional Bank) (Palmer et al., 2008) and the OntoNotes project (Hovy et al., 2006) both annotate for Arabic semantic information. Alkuhlani and Habash (2011) add annotations marking functional gender and number, and rationality; and Abdul-Mageed and Diab (2012) annotate the sentence level with sentiment labels. Tounsi et al. (2009) automatically converted the PATB to a lexical functional grammar (LFG) representation. Similarly, Habash and Roth (2009) used a similar technique to build an initial version of CATiB. We use this CATiB version of PATB to evaluate our approach in this paper. Also related to this is the work on automatic enrichment of specific features, e.g., Habash et al. (2007a) demonstrated that nominal case, can be determined for gold syntactic analyses at high accuracy. We replicate their results and improve upon them. And unlike them, we handle all the morphological features in the PATB, not just case.

Morphological Disambiguation There has been a lot of work on Arabic POS tagging and morphological disambiguation (Diab et al., 2004; Habash and Rambow, 2005; Smith et al., 2005; Hajič et al., 2005; Roth et al., 2008; Habash et al., 2013). These approaches are intended to apply to raw text and determine the appropriate in-context morphological reading for each word. In contrast, in this paper, we are starting from a partially disambiguated and relatively rich representation: we have tokenization, general POS tags and syntactic dependency information.

Finally, morphological information (beyond tokenization) has been shown to be useful for many NLP applications. Marton et al. (2011) demonstrated that morphology helps Arabic parsing. Using morphological features such as case has also improved parsing for Russian, Turkish and Hindi (Nivre et al., 2008; Eryigit et al., 2008; Nivre, 2009). Other work has shown value for morphology in the context of Arabic named entity recognition (Benajiba et al., 2009). These results support the value of our goal of enriching resources with morphological information, which then can be used to improve different NLP applications.

3 Linguistic Background

In this section, we present some relevant general linguistic facts about Arabic and then discuss the specifics of the tagsets we work with in this paper.

Arabic Linguistic Facts The Arabic language poses many challenges for NLP. Arabic is a morphologically complex language which includes rich inflectional and cliticizational morphology, e.g., the word وسيكتبونها *w+s+y-ktb-wn+hA*[1] 'and they will write it' has two proclitics, one prefix, one suffix and one pronominal enclitic. Additionally, Arabic has a high degree of ambigiouty due to the absence of the diacritics and inconsistent spelling of letters such as Alif, أ Â and Ya ي *y*. The Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2004), which is used in the PATB, produces an average of 12 analyses per word.

In this paper, we work with gold tokenized Arabic as it appears in the PATB and CATiB treebanks.

[1] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007b): (in alphabetical order) *Abtθ jHxdðrzsšSDTDˇ ς γfqklmnhwy* and the additional symbols: ' ء, Â أ, Ă إ, Ā آ, ŵ ؤ , ŷ ىء , ħ ة, ý ى.

As such, the words are partially disambiguated with regards to possible tokenizable clitics and Alif/Ya spelling forms. That said, there is still a lot of ambiguity remaining especially because diacritics are not marked. Words in the treebank may be ambiguous in terms of their POS, lemmas and inflectional features. The inflectional features include gender, number, person, case, state, mood, voice, aspect and the presence of the determiner +ال *Al+* 'the', which is not tokenized off in the treebanks.

Arabic has a well known discrepancy in form and function that appears most commonly in the form of irregular plurals, called Broken Plurals, which although functionally are plural, have singular suffixes. We will not discuss form and function discrepancy in this paper except as needed. For more on this see Habash (2010).

The Buckwalter Tagset The Buckwalter POS tagset is perhaps one of the most commonly used tagsets for Arabic NLP research. The tagset's popularity is in part due to its use in the PATB. Buckwalter tags can be used for tokenized and untokenized text. The untokenized tags are produced by BAMA (Buckwalter, 2004) and consist of 485 tags. The tokenized tags, which are used in the PATB, are derived from the untokenized tags and can reach thousands of tags. Both variants use the same basic 70 or so sub-tag symbols (such as DET 'determiner', NSUFF 'nominal suffix', ADJ 'adjective' and ACC 'accusative') (Maamouri et al., 2009a). These sub-tags are combined to form around 170 morpheme tags such as NSUFF_FEM_SG 'feminine singular nominal suffix' and CASE_DEF_ACC 'accusative definite'. The word tags are constructed out of one or more morpheme tags, e.g. DET+NOUN_PROP+CASE_DEF_NOM for the word الصين *Al+Siyn+u* 'China'.

CATiB Trees and POS Tags CATiB uses the same basic tokenization scheme used by PATB and PADT. However, the CATiB POS tagset is much smaller. Whereas in practice PATB uses 485 Buckwalter tags specifying every aspect of Arabic word morphology such as definiteness, gender, number, person, mood, voice and case, CATiB uses 6 POS tags: **NOM** (non-proper nominals including nouns, pronouns, adjectives and adverbs), **PROP** (proper nouns), **VRB** (verbs), **VRB-PASS** (passive-voice verbs), **PRT** (particles such as prepositions or conjunctions) and **PNX** (punctuation). CATiB uses a dependency representation that models predicate-argument structure (subject, object, etc.) and Arabic nominal structure (idafa, tamyiz, modification). The eight CATiB relation labels are: **SBJ** (subject of verb or topic of simple nominal sentence), **OBJ** (object of verb, preposition, or deverbal noun), **TPC** (topic in complex nominal sentences containing an explicit pronominal referent), **PRD** (predicate marking the complement in some copular constructions), **IDF** (relation between the possessor [dependent] and the possessed [head] in the idafa/possessive nominal construction), **TMZ** (relation of the specifier [dependent] to the specified [head] in the tamyiz/specification nominal constructions), **MOD** (general modifier of verbs or nouns), and **—** (marking *flatness* inside constructions such as first-last proper name sequences). This relation label set is much smaller than the twenty or so dash-tags used in PATB to mark syntactic and semantic functions. Furthermore, no empty categories, coreference, or semantic relations (e.g., TMP or LOC) are provided (Habash and Roth, 2009). A detailed dis-

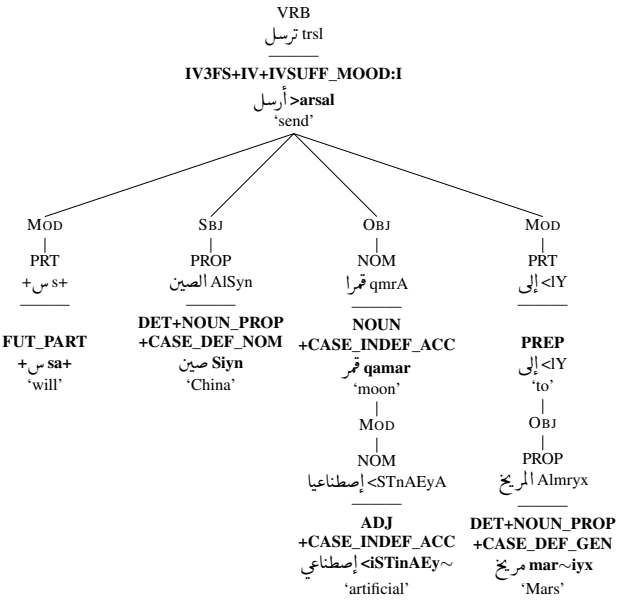

Figure 1: An example dependency tree for the sentence ستر سل الصين قمرا إصطناعيا إلى المريخ *s+trsl AlSyn qmrA ĂSTnAς yA Ălý Almryx* 'China will send a satellite to Mars'. In every tree node, the terms above the line are part of the CATiB annotations: the word, POS (VRB, PRT, PROP, NOM) and relation (MOD, SBJ, OBJ). The terms under the line are the Buckwalter POS tag, the lemma and the gloss, respectively.

cussion of CATiB guidelines and further comparison with PATB appears in (Habash et al., 2009).

In this paper, we target the enrichment of CATiB with the morphological information used in the PATB: Buckwalter POS tags and lemmas. We do not address other kinds of rich information. Figure 1 presents an example of a CATiB tree with the extensions we predict automatically for each word.

4 Approach

We define our task as assigning a Buckwalter POS tag and lemma to each word in a CATiB syntactic tree, i.e., disambiguating the CATiB POS tag in terms of the finer grained Buckwalter tag in context. Our approach utilizes a variety of corpus-based and rule-based techniques. We use corpus-based techniques that exploit available training data in the form of portions of the PATB that are automatically converted to CATiB style trees. We also use rule-based solutions that allow us to apply linguistic knowledge and insights. An important tool that we use is a morphological analyzer which generates for every word all possible out-of-context analyses. We use these analyses as a constraint on the space from which we will select the appropriate in-context tag. The approach is quite similar to how MADA (Morphological Analysis and Disambiguation for Arabic) (Habash and Rambow, 2005) works except that we are using the CATiB tree as the context in which we disambiguate. Given the degree of richness of the tree, we expect to outperform basic disambiguation on text.

In the rest of this section, we discuss our general strategy, followed by a detailed presentation of our approach: morphological disambiguation and morphological filtering.

4.1 From CATiB to Buckwalter: Devising a Strategy

In CATiB trees, different pieces of information can be relevant to different disambiguation tasks. We analyzed a sample of the data we have and obtained the following observations which we use to devise our strategy for how to address different types of ambiguity:

- Words in the CATiB treebank are tokenized. This resolves many ambiguous cases: analyses involving cliticized prepositions or conjunctions are dismissed. Further more, separated clitics are marked, which restricts their reading. The ambiguity in terms of the number of lemmas per word reduces from 2.7 for untokenized words to just 1.1 for tokenized words.
- The CATiB POS tagset, although two orders of magnitude smaller than the Buckwalter tagset, provides a lot of information. It resolves ambiguity amongst verbs (active or passive), nominals, particles, proper nouns and punctuation.
- The CATiB tags NOM and PRT are the most ambiguous. They are challenging because there are both lexical and morphosyntactic features at play. We rely on our training data to learn models of how to disambiguate them. The CATiB treebank annotation does not deterministically allow us to identify the finer grained tag using the POS and relations alone: e.g., the NOM child of an NOM parent (with the relation MOD) can be an ADJ (67% probability) or a NOUN (21%).
- Case, state, mood and to a lesser degree aspect are syntactically dependent features, for which we use the CATiB tree and linguistic rules to disambiguate the correct value in context.
- Gender, number and person are expressed using affixes that highly limit the feature-value possibilities, e.g., the suffix ـهـ +ħ deterministically selects for +NSUFF_FEM_SG suffix tag.

4.2 Morphological Analysis & Disambiguation

We use the morphological analyzer BAMA to get a list of all possible analyses for a word. BAMA returns unranked analyses for untokenized text only. Since we know that the input is already tokenized, we built an extension to BAMA that handles clitics and accepts analyses that are consistent with the tokenization of the input, discarding all other analyses.

We use both the morphological analyzer BAMA and a training set from the PATB to predict the Buckwalter tag for a given word. We use BAMA to get a list of all possible Buckwalter tag and lemma pairs for each word. We then rank these choices using one of the following two methods: a maximum likelihood estimate model (MLE) conditioned on specific features in the CATiB tree or a MADA-like suite of classifiers that select for specific POS tag features such as gender or number.

4.2.1 Maximum Likelihood Model

The MLE model ranks the set of choices from BAMA returning the most probable analysis. We consider two models:

- **MLE Baseline 1** selects the Buckwalter tag with the highest unconditioned probability in the training data, P(BW), among the set of BAMA choices for the word whose tag we want to determine.
- **MLE Baseline 2** selects the Buckwalter tag with the highest probability conditioned on the word and CATiB tag: P(BW|word,CATiB). This model backs off to the Buckwalter tag with the highest probability conditioned on the CATiB tag, i.e., P(BW|CATiB), and then backs off to MLE Baseline 1.

4.2.2 Analysis and Disambiguation of Tokenized Arabic

We retrained the MADA system (Habash and Rambow, 2005) using a tokenized version of the PATB. We call the new version TADA: Tokenized Analysis and Disambiguation of Arabic. TADA takes tokenized text, and returns a ranked list of analyses for each tokenized word and clitic. Just like MADA, TADA uses BAMA to identify possible analyses of the word. It then uses a suite of classifiers to predict inflectional and lexical features that are used to rank the possible analyses.

As expected, TADA outperforms the simple MLE models described earlier; however, its performance is not high enough since it makes no use of tree features. The results are presented in Section 5. However, we will present here a preliminary error analysis of TADA's output to motivate the morphological filters presented next (Section 4.3).

TADA Preliminary Error Analysis We considered the first 100 errors in the Buckwalter tags in our development set. About half of the errors involved a problem in case (42%), state (13%) or mood (3%). Case and state errors had many overlaps. All of these errors are syntactically determinable using the tree representation in a manner similar to Habash et al. (2007a). In 17% of the cases, a POS error can be resolved using the CATiB tag, (e.g., proper noun vs adjective or verb). In 2% of the cases, the error involved an orthographic normalization (Alif-form) that led to an undesirable solution (e.g.,ألسنة Âlsnħ 'tongues' vs السنة Alsnħ 'the-year'). These cases should be resolved by enforcing the CATiB tree word form. Ambiguity in CATiB tags was a problem for nominal forms 15% of the time (e.g., NOUN vs ADJ), particles 10% of the time (e.g., +و w+ 'and' can be CONJ or SUB_CONJ), and pronouns 5% of the time (e.g., +هم +hm 'them' can be IVSUFF_DO:3MP or PVSUFF_DO:3MP [attached to an imperfective or perfective verb]).[2] In 1% of the cases, there was an error involving ambiguity in number (dual/plural). And finally, in 3% of the cases, we determined that the gold POS tag was actually incorrect. Within the same set of sentences studied, we found 18 lemma choice errors. Almost all, except for three cases, involve a nominal form ambiguity resulting from diacrtic absence, e.g., مهدد *muhad∼id* 'threatening' or *muhad∼ad* 'threatened'. Eight of the 18 cases (or 44%) happened without an accompanying POS error. Overall, the accuracy of lemma choice is highly dependent on the correctness of the chosen core Buckwalter tag; lemma accuracy when the tag is correct is 97.9%, but it drops to 71.3% when the tag is wrong.

4.3 Morphological Filters

We implemented a set of filters that take the list of ranked analyses produced by TADA and discard any analyses that are inconsistent with the filters' decisions in the tree context. TADA ranking is preserved among the remaining analyses.

4.3.1 CATiB Filter

TADA returns all analyses for word, including different forms of the word (i.e., different Alif/Ya forms as part of BAMA's back-off mode). For example, when given the word على ʕlý, TADA returns analyses for both the words على ʕlý 'on' and علي ʕly 'Ali'. Since the input to our system is the gold word form from CATiB trees, the CATiB filter will discard analyses that do not match the given word form.

The CATiB filter also resolves some POS ambiguity given information in the CATiB POS tag. For example, the CATiB POS tags NOM or VRB can easily decide whether the ambiguous word كاتب *kAtb* is a noun (*kAtib* 'writer') or a verb (*kAtab* 'to correspond').

[2] This is a peculiarity of the tagset used in PATB. The distinction does not seem to be necessary to our knowledge, but we still consider it the gold goal.

4.3.2 Pronominal Filter

The pronominal filter (PRON) selects the pronouns that are consistent with the verbs they are attached to. A pronoun attached to a verb could either be IVSUFF_DO, PVSUFF_DO or CVSUFF_DO depending on whether the verb is imperfective (IV), perfective (PV), or imperative (CV).

4.3.3 Noun/Adjective Filter

The noun/adjective (NOUN/ADJ) filter is applied to words with the CATiB tag NOM. It uses a nominal classifier, which classifies CATiB NOM words into one of the following Buckwalter noun/adjective classes: NOUN, NOUN.VN, NOUN_QUANT, NOUN_NUM, ADJ, ADJ.VN, ADV_COMP, ADV_NUM. The NA (not-applicable) tag is assigned to all other words. For example, the classifier will decide whether كبيرة *kbyrħ* is a noun 'abomination' or an adjective 'great [feminine singular]' based on the context.

To build the nominal classifier, we use Yamcha (Kudo and Matsumoto, 2003), a support-vector-machine-based sequence tagger trained on our PATB training data. We use the following set of features: the word form, CATiB POS tag, parent features (word form, CATiB POS tag), dependency relation, order of appearance (the word comes before or after its parent), the distance between the word and its parent, and different types of relation-child POS (REL-CTB) features. The REL-CTB features state whether a word has a child with a CATiB POS (CTB) under a dependency relation (REL). A word can have 0 or more children. We have six CATiB POS tags and eight dependency relations and thus up to 48 different REL-CTB binary learning features. An example of this feature is a PRT that has a child NOM under a dependency relation OBJ. In this case, the value of the feature OBJ-NOM is 1. We also add a window of two words before and two words after the word being tagged as static features, and the tag of the previous two words as dynamic features. The nominal classifier predicts the correct nominal class with an accuracy of 97.70%.

4.3.4 Particle Filter

The particle filter (PRT) selects the specific Buckwalter POS for a particle. For example, the particle ما *mA* can be the negative particle 'not', the relative pronoun 'that' or the interrogative pronoun 'what?'. The PRT filter uses a particle classifier that uses the same learning features and training data as the nominal classifier. The particle classifier predicts the correct particle class with an accuracy of 99.54%.

4.3.5 Verbal Mood and Aspect Filter

The Buckwalter POS tags for verbs have three markers for aspect: imperfective (IV), perfective (PV), and imperative (CV); and three markers for mood: jussive (J), subjunctive (S) and indicative (I). We apply our rule-based mood-and-aspect filter (MOOD/ASPECT) to words that have the CATiB tag VRB or VRB-PASS.

If the verb is preceded by a jussive, subjunctive or future particle then it is imperfective in aspect and its mood is determined by the particle. The mood is indicative if the verb is preceded by a future particle such as سوف *swf* 'will'; it is jussive if the verb is preceded by a jussive particle such as لم *lm* 'not+past', +ل *l+* 'for'; and it is subjunctive if the verb is preceded by a subjunctive particle such as أن *Ân* 'that', لن *ln* 'not+future', كي *ky* 'so as to', and حتى *Htý* 'until'. This is also valid when a negating لا *lA* intervenes between the subjunctive particle and the verb. If the verb is proceeded with the particle لقد *lqd* 'already', then the verb is perfective. Otherwise, the verb could be either imperfective (with an indicative mood), perfective or imperative (all allowed through the filter).

4.3.6 Nominal State Filter

The Buckwalter POS tags have three nominal state markers: INDEF, DEF and POSS.[3] The nominal state filter (STATE) applies the following rules: If the word is head of an idafa (IDF), then we exclude the INDEF analyses. Otherwise, we exclude the POSS and the non-Al/DET determined DEF analysis (which are only used for IDF heads).

4.3.7 Nominal Case Filter

The nominal case filter (CASE) assigns the values *nom* (nominative), *acc* (accusative) or *gen* (genitive) to each NOM/PROP word primarily based on the CATiB dependency relation label that describes the type of relation between the word and its parent. The nominal case filter extends the case predictor in Habash et al. (2007a). The following four rules are applied in sequence.

[3] These values do not exactly match the functional values for state in Arabic (Smrž, 2007).

- RULE 1: Assign *acc* to all NOM/PROP words as a default.

- RULE 2: Assign *nom* to NOM/PROP words that (a) head the tree, (b) have the label TPC, (c) have the label SBJ but are not headed by a particle from the closed class of *Inna and its sisters* (Habash et al., 2007a), or (d) have the label PRD but is not headed by a verb or deverbal noun. Exempt words in the closed class of adverb-like nouns such as فوق *fwq* 'over', قبل *qbl* 'before', and حول *Hwl* 'around'.

- RULE 3: Assign *gen* to NOM/PROP words that have the label OBJ under a preposition, or that have the label IDF.

- RULE 4: All children of NOM/PROP parents whose label is MOD, and NOM/PROP children of conjunctions whose label is OBJ, copy the case of their parent. Conjunctions carry the case temporarily to pass on agreement.

4.3.8 MLE Override

We added an MLE-based component to override answers that are provided by our final system. We used a no-BAMA version of the MLE Baseline 2. The difference between the MLE override component and MLE Baseline 2 is that it takes into account all possible Buckwalter POS tags that appear in the training set for a specific word regardless of whether they are provided by BAMA or not. This MLE override component is trained on the same training set and returns the most common Buckwalter tag and lemma pair for a given word form and CATiB POS tag pair. BAMA is not used here since the reason behind this additional step is to overcome any limitation caused by using BAMA to start with. These limitations include primarily cases of BAMA failure to produce analyses (OOV) or minor version differences between BAMA and the PATB. If a Buckwalter tag and lemma pair appear above a threshold of n times and always with the same word-lemma pair, then we override our answer with the new answer from the MLE. When we override, we only override the core part of the Buckwalter tag. We do not override the state, case, and mood features since they are syntactic features. We tried different values for the threshold and got the best results when $n = 4$.

4.4 Putting it All Together

TADA provides an initial list of ranked analyses. Then, the morphological filters discard analyses that are not consistent with the CATiB tree information. The analysis with the highest TADA rank among the remaining analyses is selected as the answer.

We apply our filters in the following order. We first apply the CATiB filter. After that, we apply the pronominal, noun/adjective and particle filters. These three filters can be applied in any order since they are applied on disjoint sets of words. The next filter is the mood/aspect filter which has to be applied after the particle filter since it depends on the particle choice in predicting the mood of the following verb. At this point, we freeze the lemma choice for the word. The next two filters, state and case, look at syntactic features and should not affect the choice of the lemma.

We use two back-off mechanisms. The first one is with the application of each filter. If the effect of applying a filter results in an empty set (no match found) then we undo the effect of the filter and pass the list of analyses as is to the next filter. The second mechanism is using the MLE override at the end of the pipeline. TADA, the noun/adjective and particle filters, and the MLE override use corpus-based components while all other filters are rule-based.

5 Evaluation

5.1 Experimental Settings

We use a CATiB version of the PATB part 3v3.1 and part 2v3.0 released by the Linguistic Data Consortium (LDC) (Maamouri et al., 2004). We use the train/development/test (80/10/10) splits of Marton et al. (2010) for PATB part 3v3.1 (16.6K sentences; 400K tokens): we use their train as our training data, their development as the tuning data for TADA and their test as our development set. For our blind test, we use the first 1000 sentences in PATB part 2v3.0 (38K tokens).

We report all results in terms of token accuracy on the full Buckwalter tag, reduced Buckwalter tag and the lemma. The reduced Buckwalter tag is the Buckwalter tag without case, state, and mood. The number of tags is reduced to 220 tags (compared to 485 tags for the full Buckwalter tagset).

In cases of gold full Buckwalter tags that are underspecified for case, state or mood, we do not penalize our systems if our more specific predicted

	Full BW	Reduced BW	Diff	Lemma
MLE Baseline 1	57.19	73.44	16.25	90.87
MLE Baseline 2	77.69	93.27	15.58	94.31
TADA	86.15	94.04	7.89	96.97
++ CATiB	87.33	95.50	8.17	97.72
++ PRON	88.16	96.32	8.16	97.72
++ NOUN/ADJ	88.93	97.26	8.33	97.72
++ PRT	89.24	97.57	8.33	97.72
++ MOOD/ASPECT	89.46	97.60	8.14	97.74
++ STATE	89.92	97.61	7.69	97.74
++ CASE	94.90	97.61	2.71	97.74
++ MLE override	**95.27**	**98.00**	**2.73**	**97.81**

Table 1: Accuracy of enriching CATiB trees with Buckwalter (BW) tags and lemmas on the **development** set. Reduced Buckwalter is similar to Buckwalter, but ignores case, mood and state. The **Diff**erence between the two metrics highlights the errors from case, mood and state.

tag otherwise matches the gold tag. Words whose lemmas are unknown (nolemma, TBupdate) or has the lemma DEFAULT (including digits and punctuation) are excluded from the evaluation, but not training: in the development set, 4,498 out of 25,446 words were excluded ($\sim$18%).

5.2 Results

Table 1 shows the results of our experiments on the development set. Considering the baseline systems, we see that using both the CATiB POS tag and the word form in MLE Baseline 2 gives us a 20.5% absolute increase above MLE Baseline 1. Using TADA improves the performance significantly (adding 8.46% absolute over MLE Baseline 2). Every additional morphological filter has a positive impact and the improvement of the accuracy for full Buckwalter with each new filter ranged between 0.22% and 1.18% absolute except for the case filter, which adds almost 5%. Adding the MLE override has a positive impact on the accuracy of the full and reduced Buckwalter tags and the lemma.

We apply our baselines, TADA, TADA+filters and TADA+filters+MLE to the blind test set (see Table 2). The test set is a bit harder than the development set, but the results are consistent with those seen for the development set.

5.3 Error Analysis

We conducted an analysis of the errors in the output of the final system TADA+filters+MLE on the development set. We considered 100 randomly selected error cases and examined them in the CATiB trees

	Full BW	Reduced BW	Diff	Lemma
MLE Baseline 1	55.96	71.88	15.92	90.77
MLE Baseline 2	77.15	92.88	15.73	94.03
TADA	86.49	94.42	7.93	96.63
++ All filters	93.44	97.25	3.81	97.13
++ MLE override	**93.61**	**97.43**	**3.82**	**97.17**

Table 2: Accuracy of enriching CATiB trees with Buckwalter (BW) tags and lemmas on the **blind test** set.

to assess the source of the error. About 37% of all errors are due to gold treebank errors: 21% are gold tree structure/relation errors and 16% are gold POS errors. The rest of the errors result from failures in our system. The most common error is in NOM disambiguation: NOUN/NOUN_NUM, NOUN/ADJ, NOUN/NOUN_QUANT, etc. The NOM errors accounted for 33% of all errors. Case comes second with 12% errors, then PRT and gender-number-person errors with 5% each. State errors contribute to 3% of total errors.

5.4 Learning Curve Study

The non-rule-based components of our approach, namely TADA, NOUN/ADJ and PRT filters, and MLE override depend on the existence of an annotated treebank in rich format. To understand the degree of dependence, we ran a series of experiments on different sizes of the training data: $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, and $\frac{1}{32}$ of the full training set (341.1K words). These data sets were used to train new versions of TADA, the NOUN/ADJ and PRT filters, and the MLE override. The results of running TADA and the final system on the development set using the different data sets are summarized in Tables 3 and 4, respectively. As expected, when the training data size goes down the accuracy goes down. Our final system, which adds filters on top of TADA, had a significant effect on the performance as shown in Table 4. Using only 10.6K of annotated words, the quality of TADA reduces sharply (12.12% absolute reduction in accuracy) while the overall effect on our full system is a lot smaller (2.03% absolute drop). Similarly, the performance of the nominal and particle classifiers degrade when trained on less data. When we use $\frac{1}{32}$ of the training data, the correct nominal class is predicted at an accuracy of 90.49% (7.21% absolute drop), while the correct particle class is predicted at an accuracy of 96.59% (2.95% absolute drop). We used the MLE override threshold determined based

	Size	Full BW	Reduced BW	Diff	Lemma
1/32	10.6K	74.03	88.78	14.75	93.41
1/16	21.3K	77.16	90.30	13.14	94.37
1/8	42.6K	79.76	91.63	11.87	95.56
1/4	85.3K	81.91	92.80	10.89	96.22
1/2	170.7K	84.12	93.62	9.50	96.74
1	341.1K	86.15	94.04	7.89	96.97

Table 3: Accuracy of enriching CATiB trees with Buckwalter (BW) tags and lemmas using TADA only for different training sizes on the development set.

	Size	Full BW	Reduced BW	Diff	Lemma
1/32	10.6K	93.24	95.81	2.57	95.68
1/16	21.3K	93.67	96.28	2.61	96.27
1/8	42.6K	94.14	96.79	2.65	96.94
1/4	85.3K	94.56	97.26	2.70	97.22
1/2	170.7K	94.96	97.66	2.70	97.61
1	341.1K	95.27	98.00	2.73	97.81

Table 4: Accuracy of enriching CATiB trees with Buckwalter (BW) tags and lemmas using our best performing system for different training sizes on the development set.

on the full training data, which may not be optimal for smaller data sets.

The contribution of our full system over TADA when using $\frac{1}{32}$ of the full training data is over 19% absolute (on full Buckwalter tag determination) compared to 9% when using the full training data. The morph analysis (out of context) is the same for all experiments and that this provides a lot of stability to the results. The high lemma accuracy overall is a result of disambiguating tokenized words, where the average numbers of lemmas per word is only 1.1 as mentioned above. These results suggest that our approach is usable even in the early stages of developing new richly annotated treebanks.

5.5 Extrinsic Evaluation

We applied our automatic enrichment to the underspecified CATiB treebank (as opposed to the parts of PATB, which we used throughout the paper to simulate CATiB). We evaluate the added value of these annotations by using them to extend the training data for the morphological tagger MADA (Habash and Rambow, 2005), which is used on untokenized text. We train a new set of MADA classifier models using a combination of the original MADA (v 3.2) training data (578K words taken from PATBs 1, 2 and 3) and the enriched CATiB data (218K words). We apply the new MADA system to our development set and evaluate on several metrics. As a baseline, we process the same development set using MADA (v 3.2). Other than the training data used to construct the classifier models, there are no differences between the two systems. The CATiB-enriched system results in a Buckwalter POS tag accuracy of 85.6% (a 2.2% error reduction over the baseline). When evaluating on the set of 14 MADA morphological features, the new system results in a 85.7% accuracy (2.4% error reduction). The new system also improves PATB segmentation accuracy (99.2%, a 5.4% error reduction). In the future, we will evaluate the contribution of the additional annotations in the context of other applications, such as syntactic parsing.

6 Conclusion and Future Work

We have demonstrated that an underspecified version of an Arabic treebank can be fully specified for Arabic's rich morphology automatically at an accuracy rate of 94%-95% for POS tags and 97% for lemmas. Our approach combines a variety of techniques from corpus-based statistical models (which require some rich annotations) to linguistic rules that target specific phenomena. Since the underspecified treebank is much faster to manually annotate than its fully specified version, these results suggest that the cost of treebanking can be reduced by designing underspecified treebanks that can be subsequently enriched automatically.

In the future, we plan to extend the automatic enrichment effort to include more complex features such as empty nodes and semantic labels. We also plan to take the insights from this effort and apply them to treebanks of other languages. A small portion of a treebank that is fully annotated in rich format will of course be needed before we can apply these insights to other languages.

Acknowledgments

The first author was funded by a scholarship from the Saudi Arabian Ministry of Higher Education. The rest of the work was funded under DARPA projects number HR0011-08-C-0004 and HR0011-12-C-0014. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

M. Abdul-Mageed and M. Diab. 2012. AWATIF: A Multi-Genre Corpus for Modern Standard Arabic Subjectivity and Sentiment Analysis. *The 8th International Conference on Language Resources and Evaluation (LREC2012)*.

Sarah Alkuhlani and Nizar Habash. 2011. A Corpus for Modeling Morpho-Syntactic Agreement in Arabic: Gender, Number and Rationality. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, Portland, Oregon, USA.

Yassine Benajiba, Mona Diab, and Paolo Rosso. 2009. Arabic Named Entity Recognition: A Feature-driven Study. *IEEE Transactions on Audio, Speech & Language Processing*, 17(5):926–934.

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

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In *Proceedings of the 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, Boston, MA.

Kais Dukes and Tim Buckwalter. 2010. A Dependency Treebank of the Quran using Traditional Arabic Grammar. In *Proceedings of the 7th international conference on Informatics and Systems (INFOS 2010)*, Cairo, Egypt.

Gülsen Eryigit, Joakim Nivre, and Kemal Oflazer. 2008. Dependency Parsing of Turkish. *Computational Linguistics*, 34(3):357–389.

Nizar Habash and Owen Rambow. 2005. 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'05)*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitch Marcus. 2007a. Determining Case in Arabic: Learning Complex Linguistic Behavior Requires Complex Linguistic Features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1084–1092.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007b. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Jan Hajič, Otakar Smrž, Tim Buckwalter, and Hubert Jin. 2005. Feature-based tagger of approximations of functional Arabic morphology. In *Proceedings of the Workshop on Treebanks and Linguistic Theories (TLT)*, Barcelona, Spain.

Erhard Hinrichs, Sandra Kübler, Karin Naumann, Heike Telljohann, and Julia Trushkina. 2004. Recent Developments in Linguistic Annotations of the TüBa-D/Z Treebank. In *Proceedings of the Third Workshop on Treebanks and Linguistic Theories*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA.

Taku Kudo and Yuji Matsumoto. 2003. Fast Methods for Kernel-Based Text Analysis. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of ACL*, pages 24–31.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank : Building a Large-Scale Annotated Arabic Corpus.

Mohamed Maamouri, Ann Bies, Sondos Krouna, Fatma Gaddeche, and Basma Bouziri, 2009a. *Penn Arabic Treebank Guidelines*. Linguistic Data Consortium.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2009b. Creating a Methodology for Large-Scale Correction of Treebank Annotation: The Case of the Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2010. Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 13–21, Los Angeles, CA, USA, June.

Yuval Marton, Nizar Habash, and Owen Rambow. 2011. Improving Arabic Dependency Parsing with Form-based and Functional Morphological Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, Portland, Oregon, USA.

Henrik Müller. 2010. Annotation of Morphology and NP Structure in the Copenhagen Dependency Treebanks (CDT). In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*, pages 151–162.

Joakim Nivre, Igor M. Boguslavsky, and Leonid L. Iomdin. 2008. Parsing the SynTagRus Treebank of Russian. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 641–648, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre. 2009. Parsing Indian Languages with MaltParser. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 12–18.

Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christoper D. Manning. 2002. LinGO Redwoods - A Rich and Dynamic Treebank for HPSG. In *LREC workshop on parsing evaluation*, Las Palmas, Spain.

Martha Palmer, Olga Babko-Malaya, Ann Bies, Mona Diab, Mohamed Maamouri, Aous Mansouri, and Wajdi Zaghouani. 2008. A Pilot Arabic Propbank. In *Proceedings of LREC*, Marrakech, Morocco, May.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. In *Proceedings of ACL-08: HLT, Short Papers*, pages 117–120, Columbus, Ohio.

Noah Smith, David Smith, and Roy Tromble. 2005. Context-Based Morphological Disambiguation with Random Fields. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP05)*, pages 475–482, Vancouver, Canada.

Otakar Smrž and Jan Hajič. 2006. The Other Arabic Treebank: Prague Dependencies and Functions. In Ali Farghaly, editor, *Arabic Computational Linguistics: Current Implementations*. CSLI Publications.

Otakar Smrž, Viktor Bielický, Iveta Kouřilová, Jakub Kráčmar, Jan Hajič, and Petr Zemánek. 2008. Prague Arabic Dependency Treebank: A Word on the Million Words. In *Proceedings of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Marrakech, Morocco.

Otakar Smrž. 2007. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University in Prague, Prague, Czech Republic.

Lamia Tounsi, Mohammed Attia, and Josef van Genabith. 2009. Automatic Treebank-Based Acquisition of Arabic LFG Dependency Structures. In *Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages*, pages 45–52, Athens, Greece.