Diacritic Annotation in the Arabic Treebank and Its Impact on Parser Evaluation

Mohamed Maamouri, Seth Kulick, Ann Bies

Linguistic Data Consortium
University of Pennsylvania
{maamouri,skulick,bies}@ldc.upenn.edu

Abstract

The Arabic Treebank (ATB), released by the Linguistic Data Consortium, contains multiple annotation files for each source file, due in part to the role of diacritic inclusion in the annotation process. The data is made available in both ”vocalized” and ”unvocalized” forms, with and without the diacritic marks, respectively. Much parsing work with the ATB has used the unvocalized form, on the basis that it more closely represents the ”real-world” situation. We point out some problems with this usage of the unvocalized data and explain why the unvocalized form does not in fact represent ”real-world” data. This is due to some aspects of the treebank annotation that to our knowledge have never before been published.

1. Introduction

The Arabic Treebank (ATB), released by the Linguistic Data Consortium, contains multiple annotation files for each source file. This is due in part to the role of diacritic inclusion in the annotation process, in which diacritic marks that are usually absent from the written text are included in the annotation along with the Part-of-Speech tags and syntactic structure. The data is made available in both ”vocalized” and ”unvocalized” forms, with and without the diacritic marks, respectively. Much parsing work with the ATB has used the unvocalized form, on the basis that it more closely represents the ”real-world” situation.

The purpose of this paper is twofold. First, we discuss some problems with this usage of the unvocalized data by parsers and explain why the unvocalized form does not in fact represent ”real-world” data. This is due to some aspects of the treebank annotation that to our knowledge have never before been published. Second, we discuss why diacritics cause an Arabic text NLP pipeline to be more complex than an English NLP pipeline, and we describe how we have modified the ATB corpus to better reflect this complexity, offering researchers the maximum flexibility for experimentation with different types of parser input.

There is now extensive documentation included with the release covering these issues and other details.

2. Overview of ATB Annotation and the Role of Diacriticization

Arabic is a highly inflected language. It has many different word forms for any given lemma, which include the root, its internal structure, prefixes, suffixes and clitics. One reason for this ambiguity is the absence of diacritic markers in most written text, such as the ATB newspaper sources. These diacritics include representations of short vowels, gemination, and definiteness marking. The marking of the presence of the glottal stop (hamza) is also sometimes left out in the case of initial alif. For convenience, we will also use the terms ”vocalized” and ”unvocalized” for the forms with and without the diacritic marks, respectively.

The role of diacritic annotation in the ATB is discussed in (Maamouri and Bies, 2004) but for our purposes the crucial points are these:

1. The source text consists of words that are treated as whitespace-delimited tokens. (We call these the ”source” tokens.) As noted above, these tokens are usually lacking diacritic information.

2. The ATB uses a level of annotation more accurately described as morphological analysis than as part-of-speech tagging. The source token is run through the Buckwalter Arabic Morphological Analyzer (Buckwalter, 2004). This provides all possible analyses of the token, each analysis consisting of (1) breaking a word into its various segments (prefixes, suffixes, gender and Case endings) and (2) providing a vocalization for each segment, with the diacritics appropriate for that solution. The annotator picks one such analysis as the proper solution for this token. We refer to this solution as the POS token, which includes the breakdown of the source token into its various segments, each one vocalized.

   For example, Figure 1 shows a sample output of the Buckwalter analyzer, excerpting from its complete output for the input token `ktb` كتب. In our terminology, `ktb` is the source token. The annotator picks one of the possible solutions, and this solution would be what we call the POS token. For example, `katab/PV+a/PVSUFF_SUBJ:3MS` and `kutib/PV_PASS+a/PVSUFF_SUBJ:3MS` are the first two potential POS tokens.

3. Depending on the solution, the POS token may be split

up into different tokens. Clitics that play a role in the syntactic structure are split off into separate tokens (e.g., object pronouns cliticized to verbs, cliticized prepositions, etc.).

Again continuing the example, none of the possible solutions (and therefore possible POS tokens) in Figure 1 contain segments that get split off,since Case suffixes and agreement morphemes do not get split. In Section 3.1. however we will see some examples that do get split.

4. We refer to the tokens resulting from this splitting process as the ”Treebank” tokens. These are the tokens that are used for treebank annotation. Of course, it is often the case that a POS token does not have to be split, and the corresponding Treebank token is identical to the POS token, which in turn is identical to the source token, except for the added diacritic information.

5. The treebanking proceeds using these Treebank tokens. The trees have until recently been released in two forms - ”without-vowel” and ”with-vowel”, without documentation explaining how the ”without-vowel” trees are created. The ”with-vowel” trees are the syntactic trees with tokens that are the Treebank tokens with full diacriticization. It appears to have been assumed, not unreasonably, by some users of the ATB that the ”without-vowel” trees contain tokens as they appeared in the original text, with the addition of the splitting off of clitics as done to make the Treebank tokens. The reality is actually more complex, as described in Section 3.1., with implications for what parsing evaluation should use for parser input.

3. Parser Evaluation

The first results on parsing the ATB were presented in (Bikel, 2004). This work was based on data available early in the ATB project, from what is now the first part of ATB, the AFP section. More recent results were presented in (Kulick et al., 2006) and (Maamouri et al., 2006), both of which used the ANNAHAR data (the third section). Our focus here is not with the actual parsing results, but rather the framework used for the evaluation.

3.1. Unvocalized data and ”real-world” parsing

The first parsing work faced an immediate question - should the unvocalized or vocalized form of the corpus be used? There were potential arguments for either choice:

unvocalized - In any Arabic NLP pipeline for processing text, it must be assumed that the input to the overall pipeline is unvoweled data, for the simple reason that real-life data does not have the diacritic information. Therefore, one might decide to use the unvoweled data as input to the parser because, as (Bikel, 2004) put it, ”that would ultimately be necessary for any real-world parser”, and that is indeed the choice that (Bikel, 2004; Kulick et al., 2006) made.

```
<solution>
  <lemmaID>katab-u_1</lemmaID>
  <voc>kataba</voc>
  <pos>katab/PV+a/PVSUFF_SUBJ:3MS</pos>
  <gloss>write + he/it [verb]</gloss>
</solution>
<solution>
  <lemmaID>katab-u_1</lemmaID>
  <voc>kutiba</voc>
  <pos>kutib/PV_PASS+a/PVSUFF_SUBJ:3MS</pos>
  <gloss>be written/be fated/be destined +
         he/it [verb]</gloss>
</solution>
<solution>
  <lemmaID>kitAb_1</lemmaID>
  <voc>kutub</voc>
  <pos>kutub/NOUN</pos>
  <gloss>books</gloss>
</solution>
<solution>
  <lemmaID>kitAb_1</lemmaID>
  <voc>kutubu</voc>
  <pos>kutub/NOUN+u/CASE_DEF_NOM</pos>
  <gloss>books + [def.nom.]</gloss>
</solution>
<solution>
  <lemmaID>kitAb_1</lemmaID>
  <voc>kutuba</voc>
  <pos>kutub/NOUN+a/CASE_DEF_ACC</pos>
  <gloss>books + [def.acc.]</gloss>
</solution>
<solution>
  <lemmaID>kitAb_1</lemmaID>
  <voc>kutubi</voc>
  <pos>kutub/NOUN+i/CASE_DEF_GEN</pos>
  <gloss>books + [def.gen.]</gloss>
</solution>
<solution>
  <lemmaID>kitAb_1</lemmaID>
  <voc>kutubN</voc>
  <pos>kutub/NOUN+N/CASE_INDEF_NOM</pos>
  <gloss>books + [indef.nom.]</gloss>
</solution>
<solution>
  <lemmaID>kitAb_1</lemmaID>
  <voc>kutubK</voc>
  <pos>kutub/NOUN+K/CASE_INDEF_GEN</pos>
  <gloss>books + [indef.gen.]</gloss>
</solution>
<x_solution>
  <voc>ktb</voc>
  <pos>ktb/NOUN_PROP</pos>
  <gloss>NOT_IN_LEXICON</gloss>
</x_solution>
<x_solution>
  <voc>katb</voc>
  <pos>ka/PREP+tb/NOUN_PROP</pos>
  <gloss>like/such as + NOT_IN_LEXICON</gloss>
</x_solution>
```

Figure 1: Excerpt from output of Buckwalter analyzer for ktb كتب

vocalized - However, it is not necessarily the case that the unvocalized data would be the input to the parser in a ”real-world” situation. There could be a preprocessing step inserting some or all of the missing diacritics, analogous to a Part-of-Speech tagger providing tags before input to the parser. So it is also a reasonable choice to construct parser experiments with the vocalized data. The role of diacritics in an NLP pipeline that includes parsing is very much an open question.

While this second choice is also reasonable, it is very likely that the parsing work referred to did not have access to an independent module for diacritic inclusion, and without such a system it becomes much harder to evaluate how a parser would work in an actual NLP pipeline.

Our concern here is not to argue for using either of the two alternatives. Our concern instead is to point out and explain why it is that *even the unvocalized data is not an accurate representation of what the ”real” data looks like*.

The reason for this is that for words that get split as part of the cliticization process, what has been released as the unvocalized form is not taken from the original source text file, but rather is just the vocalized form with short vowels stripped out. Because the vocalized form can also include certain types of orthographic normalization not present in the original source, it is sometimes the case that the concatenation of the unvocalized form of treebank tokens is not the same as the original source token that was broken up into those treebank tokens. (For words that do not get split, the unvocalized form is what one would expect - the word as it appears in the original source text file.)

Therefore, work that uses the unvocalized information (e.g., parsing on the unvocalized trees) is actually using a sort of hybrid data, which is neither a faithful representation of the original text data, nor the complete result of diacritic inclusion.

We give here two examples of why this can be the case.

- The white-space delimited input string (source token) is `llqDA'`[1] للقضاء . The `l` ل is a prefix for the preposition ”li” لِ, and the chosen solution from the morphological analyzer gives the POS token `li/PREP + Al/DET + qaDA'/NOUN`. The preposition is split off, creating two (vocalized) treebank tokens: `li` لِ and `AlqaDA'` اَلقَضَاء. Note that in addition to the insertion of the short vowels `i` in the first word and `a` in the second word, the second word also has the consonant `A`.[2]

  The unvocalized forms of the tokens are `l` and `AlqDA'`, resulting from stripping out the short vowels. The point is that the unvocalized token `AlqDA'` contains an initial alif `A` that is not included in the source text. The unvocalized token therefore contains disambiguating information that is not present in the source text.

- The source token is `Aly` الى. The vocalized solution for the POS token is `<iläy/PREP+˜a/PRON_1S`, in which there are two segments, `<ilay/PREP` and `˜a/PRON_1S`. The pronoun is split off, creating two (vocalized) treebank tokens.

  Focusing just on the PREP, the vocalized treebank token is `<ilay`, which not only adds the short vowels `i` and `a`, but also corrects for the ”missing hamza” problem by normalizing `A` to < The unvocalized form of this token results from stripping out the short vowels, and so is `<ly`. This is however not what is included in the original text file, which did not have the correct hamza placement, with `A` instead of <.

Out of 355870 instances of tokens in third section of the ATB (the section which has recently been used for parsing experiments), 13298 (3.7%) have this discrepency between the unvocalized form and the text that is actually included in the original source file. In the current release of the ATB part 3, we have significantly expanded the documentation to make all these facts clear, and to also provide a representation of the unvocalized form that is a more accurate representation of the contents of the original source file.

3.2. Interaction between the unvocalized and vocalized forms

The question of what the input to the parser should be is actually considerably more complex than just the relatively simple decision of using unvocalized or vocalized data.

There are several aspects to data processing that need to occur within an Arabic NLP pipeline: tokenization, diacritic inclusion, Part-of-Speech tagging, and parsing.[3] It is likely that some of these steps should be integrated together, and work has been done along these lines (e.g., (Habash and Rambow, 2005)), and depending on how this is done, the parser input might be expected to have tokenization but not diacritics, or only some subset of the diacritics, etc. This is in contrast to English, for which tokenization is trivial and there is of course no issue with diacriticization, and so

[1] We are using the Buckwalter transliteration scheme.

[2] This is because the sequence `lAl` is written as `ll` in Arabic orthography.

[3] And diacritic inclusion should itself probably be broken into subtasks for word-internal diacritics or diacritics representing Case or Mood information. We are abstracting away from this issue in this abstract.

the only issue for parser evaluation is whether to use gold Part-of-Speech tags or the output of a POS tagger.

While the nature of the parser input for Arabic is a matter for empirical investigation, we have modified the release format of the ATB to allow maximum flexibility for experimentation with different inputs to the parser, by making more explicit the links between the unvocalized and vocalized trees. We have done this in two different ways:

1. The text files that include information about the tokens before and after the splitting of the original tokens now both include pointers to the original source file. This allows the use of this standoff annotation for relating the different levels.

2. In addition to the trees with unvocalized and vocalized tokens, we now include trees in which the terminals are complex items, including the original token from the source file, both the unvocalized and vocalized forms, the lemma, and the gloss, thereby bringing together information that was previously scattered across different files.

4. Acknowledgements

We would like to thank Ryan Gabbard, Mitch Marcus, and all the members of the Arabic Treebank team for helpful discussions.

This work was supported in part by the Defense Advanced Research Projects Agency, GALE Program Grant No. HR0011-06-1-0003. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

5. References

Daniel M. Bikel. 2004. *On the Parameter Space of Lexicalized Statistical Parsing Models*. Ph.D. thesis, Department of Computer and Information Sciences, University of Pennsylvania.

Tim Buckwalter. 2004. Arabic morphological analyzer version 2.0. LDC2004L02. Linguistic Data Consortium.

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, June. Association for Computational Linguistics.

Seth Kulick, Ryan Gabbard, and Mitchell Marcus. 2006. Parsing the Arabic Treebank: Analysis and improvements. In *Proceedings of TLT 2006*. Treebanks and Linguistic Theories.

Mohamed Maamouri and Ann Bies. 2004. Developing an arabic treebank: Methods, guidelines, procedures, and tools. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 2–9, Geneva, Switzerland, August 28th. COLING.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A challenge to Arabic treebank annotation and parsing. In *Proceedings of the British Computer Society Arabic NLP/MT Conference*, London, UK, October.