Consistent and Flexible Integration of Morphological Annotation in the Arabic Treebank

Seth Kulick, Ann Bies, Mohamed Maamouri
Linguistic Data Consortium
University of Pennsylvania
3600 Market Street, Suite 810
Philadelphia, PA 19104 USA
E-mail: {skulick,bies,maamouri}@ldc.upenn.edu

Abstract

Complications arise for standoff annotation when the annotation is not on the source text itself, but on a more abstract representation. This is particularly the case in a language such as Arabic with morphological and orthographic challenges, and we discuss various aspects of these issues in the context of the Arabic Treebank. The Standard Arabic Morphological Analyzer (SAMA) is closely integrated into the annotation workflow, as the basis for the abstraction between the explicit source text and the more abstract token representation. However, this integration with SAMA gives rise to various problems for the annotation workflow and for maintaining the link between the Treebank and SAMA. In this paper we discuss how we have overcome these problems with consistent and more precise categorization of all of the tokens for their relationship with SAMA. We also discuss how we have improved the creation of several distinct alternative forms of the tokens used in the syntactic trees. As a result, the Treebank provides a resource relating the different forms of the same underlying token with varying degrees of vocalization, in terms of how they relate (1) to each other, (2) to the syntactic structure, and (3) to the morphological analyzer.

1. Introduction

The Arabic language presents several challenges for both annotation and natural language processing. This is in part due to its morphological characteristics, in which multiple units of meaning are combined together in a single whitespace-delimited token from the source text, which we call here a "source token." For example, a source token such as "ktbh"[1] كتبـــه might be analyzed as two parts, in which the "ktb" is a noun meaning "books" and "h" is a possessive pronoun. There are many reasons why it is important to be able to refer to these individual components separately, such as using them as leaves in a tree structure for syntactic annotation, or for word alignment with another language, such as English, in which the noun and possessive pronoun (in this case) would already be separate tokens.

Another reason why Arabic is a particular challenge is the orthographic convention of leaving out short vowels and other distinguishing marks in written text. This causes a great increase in ambiguity for resolving the analysis of each source token, as has long been noted for natural language processing. However, perhaps less noted has been the fact that this also raises many issues for annotation projects that analyze Arabic text from the source tokens to a full analysis including morphological analysis and syntactic structure. The tree structure is built not upon the source tokens, but rather on a more abstract representation, not explicitly present in the original text. This more abstract representation includes the morphological analysis and the separation of the source tokens into multiple parts, if necessary. These multiple levels of representation must all be related to each other throughout the annotation and in the end product, allowing corrections at one level to flow back to previous levels.

These issues apply to treebanks more generally. For any treebank in which the annotation is not on the explicit source text, but on some more abstract, articulated representation, similar issues will arise in some form. Even for a morphologically simple language such as English, this problem arose with the Penn Treebank, with the segmentation of "won't" into "wo" and "n't" (Bies et. al., 1995). The tree annotation continued as if the former was really "will", although the lexical item was not explicitly "normalized" to "will". In that case such needed abstractions from the data were limited enough so that the entire problem of carefully defining what the object of annotation was, and how it related to the source data, could be side-stepped. In other languages, however, this is not possible, when such abstractions are more pervasive.

These multiple levels of representation in turn become research topics for natural language processing. Given multiple levels of representation to be recovered, such research is concerned with the question of what is the most appropriate way to partition the work in a pipeline to recover the data. It is not necessarily the case that the same partition of the work as is done in the annotation is appropriate for NLP pipelines. For example, while the

[1] Throughout this paper we use the Buckwalter transliteration http://www.qamus.org/transliteration.htm

annotation project might build the syntactic structure on top of the more abstract, fully morphologically analyzed representation of the leaf tokens, users may wish to move from the original text to a more limited analysis, sufficient for the tree annotation. It is again therefore crucial that the annotation project make available the multiple levels of representation of the tokens in a consistent and accessible way so that NLP pipelines can constructively take advantage of this information.

Similarly, any treebank which utilizes a morphological analyzer to mediate between the source text and the more abstract representations needed for annotation must consider the proper way to maintain the linkage, and what to do if such an analyzer is not complete.

In this paper we discuss these issues in the context of the annotation in the Arabic Treebank at the Linguistic Data Consortium (Maamouri, Bies & Kulick 2009) (ATB). The ATB closely integrates the Standard Arabic Morphological Analyzer[2] (SAMA) into the annotation procedure, in which the morphological analysis for a token in the Treebank is selected from among the SAMA alternative solutions for that token. The ATB release provides, in addition to the syntactic structure, valuable data for machine learning experiments on the problem of relating a source text token to the correct morphological analysis, mediated through the list of possible SAMA solutions.

However, in earlier releases of the ATB there have been two problematic aspects to this interaction between the Treebank and SAMA:

1. The Treebank contained tokens that had no SAMA solution, or a solution inconsistent with SAMA. This caused problems both for the annotation workflow, which required morphological analysis before the syntactic annotation could proceed, and also in the final product, which had an undetermined amount of inconsistency with SAMA solutions.

2. While the Treebank provides, as a result of the integration with SAMA, vocalized solutions for each token, users may also want to use variants of these tokens with differing degrees of vocalization that more directly relate to the original source text. While such forms have been provided in the past, their definitions were not clear and included systematic errors.

In this paper we describe a number of changes to our workflow that solve these problems.

Regarding the first problem, tokens that cannot be annotated with a SAMA solution now receive one of two new types of entries, which is either an analysis parallel to one already included in SAMA or a minimal analysis sufficient to allow clitics to be split and syntactic annotation to proceed. In addition, the interaction between SAMA and the Treebank is now evaluated throughout the workflow, so that the link between the Treebank and SAMA is as consistent as possible.

As a result, we are now able to include information making precise the consistency of each token in the Treebank with SAMA. We expect that making such information explicit at the token level will be of great value for researchers working on the problem of determining the morphological analysis for a source token. This aspect of the work is discussed in Section 3.

For the second problem, we are now providing explicit variants of each vocalized token which include differing degrees of vocalization that more directly relate to the original text. The two such alternative forms of the tokens, "UNVOCALIZED" and "INPUT_STRING," were previously not clearly defined and contained numerous errors. To address this issue, we have implemented a new step at the end of the annotation process which addresses the problem of partitioning a source token into subsequences based on the vocalized tokens arising from that source token. Section 4 describes our solution to this problem.

2. Background Information on ATB Tokens

There are two main parts to the Arabic Treebank annotation that affect the form of the tokens.

1. The source text is broken up into roughly whitespace delimited tokens, called the "source tokens." These are the tokens that are run through SAMA, resulting in a vocalized form.

2. These source tokens are split apart if appropriate during annotation (prepositional clitics, direct object clitics, etc.). These tokens will be referred to as the "tree tokens," since these are the tokens actually used for syntactic analysis.

For all of the source tokens that receive solutions from SAMA, the syntactic annotation takes place on a partition of this solution from SAMA. The solution from SAMA is a sequence of segments, each including `[vocalization, Part-of-Speech, gloss]` information, and the sequence of such segments is partitioned into one or more tree tokens that together correspond to the original source token.

For example, the original source token "ktbh" كتبـــه has the following SAMA solution:

[2] Standard Arabic Morphological Analyzer (SAMA) Version 3.1. LDC Catalog Number: LDC2009E73.

- [kutub, NOUN, books]
- [i, CASE_DEF_GEN, def.gen]
- [hi, POSS_PRON_3MS, its/his]

The source token has received an analysis in which it has three morphological segments (one on each line here), and each segment has a vocalization (first component), part-of-speech tag (second component), and gloss (third component).

This then is the result of the "POS/morphological" level of analysis and annotation. For the syntactic annotation, this solution is partitioned into two tokens:

- [kutub+i, NOUN+CASE_DEF_GEN, books+def.gen.] كُتُب
- [hi, POSS_PRON_3MS, its/his] هِ

The reason for this is that it is these two tree tokens that contain the syntactic/semantic material appropriate for tasks such as annotation syntactic structure or word alignment. Roughly speaking, segments consisting of only inflectional material (such as the definite genitive marker here) do not become tree tokens on their own.

Annotation work with these tokens after this point within the Treebank references these vocalized forms of the tokens. That is, the syntactic structure uses "kutub+i" as a token and "hi" as a token. This is because these are the forms that are output from SAMA, and available for further annotation.

2.1 Token-Related Field Definitions

This data is represented in the Treebank in several ways, due to the different levels of annotation. In particular, there are text files listing the tokens at the two different levels of annotation, but with information relating one to the other. We call these two different text files here the "pos-level" and "treebank-level."[3]

For example, the "pos-level" file would have for the above example:

```
INPUT_STRING: كتبه
    IS_TRANS: ktbh
       INDEX: P22W10
     OFFSETS: 42-46
      TOKENS: P22W13-P22W14
      STATUS: 1
       LEMMA: [kitAb_1]
 UNSPLITVOC: (kutubihi)
         POS: NOUN + CASE_DEF_GEN
              +POSS_PRON_3MS
         VOC: kutub+i+hi
       GLOSS: books + [def.gen.]
              +its/his
```

INPUT_STRING is the actual Arabic in utf-8, and IS_TRANS is the Buckwalter transliteration of that Arabic, used throughout the Treebank.[4] INDEX is a simple paragraph/word index to the token (paragraph 22, word #10). LEMMA (a lemma for the source token, assigned by SAMA) and UNSPLITVOC are two additional pieces of information provided by SAMA, which are not the concern of this paper and so are not further discussed here. STATUS is explained in Section 3. OFFSETS refers to the character offsets in the original source text which contains the string ktbh. The field TOKENS contains the indices of the tree tokens that arise from this source token. (Note that the numbering of source tokens and tree tokens is different. This is source token #10 in paragraph 22, while it maps to tree tokens #13 and #14 in paragraph 22.) POS, VOC, and GLOSS duplicate the information in the tree tokens, simply by concatenating them together, since for users working with only the morphological annotation, and not the trees, it is more convenient to have the information in one place.

To continue this example, the "treebank-level" file, with the tree tokens, has the corresponding two entries:

```
INPUT_STRING: كتب
    IS_TRANS: ktb
     COMMENT: [Separated]
       INDEX: P22W13
     OFFSETS: 42,45
UNVOCALIZED: ktb
  VOCALIZED: kutub+i-
         POS: NOUN+CASE_DEF_GEN
       GLOSS: books + [def.gen.]

INPUT_STRING: ه
    IS_TRANS: h
     COMMENT: []
       INDEX: P22W14
     OFFSETS: 45,46
UNVOCALIZED: h
  VOCALIZED: -hi
         POS: POSS_PRON_3MS
       GLOSS: its/his
```

[3] In the actual release these files are called, somewhat awkwardly, "before-treebank" and "after-treebank."

[4] In this paper we use the terms INPUT_STRING and IS_TRANS interchangeably.

These are the two tree tokens at indices P22W13 and P22W14. The concatenation of the POS, VOC, and GLOSS information in these tokens is the same as the corresponding fields in the "pos-level" source token info. The VOCALIZED field is simply the appropriate segment(s) of the vocalized form provided by SAMA.

Note however that there are two additional forms of the tree tokens provided in the "treebank-level" file. One is the IS_TRANS, which is a partition of the IS_TRANS from the source token, such that the partition seems appropriate. For example, here the "h" in "ktbh" is clearly associated with the tree token with the vocalized form "hi", while the "ktb" belongs with the first tree token. The OFFSETS field likewise corresponds to this partition of the IS_TRANS. There is an additional field provided, the UNVOCALIZED, which does not have the additional vocalization information (short vowels "u" and "i" here), and is in fact identical to the IS_TRANS in this example.

These fields are provided for the purposes of users who wish to use the tree information but without using the full morphological analysis as the basis of the tree tokens. This is because they might wish to use an alternate pipeline which might, for example, produce only a limited tokenization and partial morphological analysis, where the tokenization is sufficient to produce the IS_TRANS forms of the tree tokens. The tree annotation refers to the tree tokens, but it is now an easy matter to use any of these alternate forms of the tree tokens (i.e., IS_TRANS or UNVOCALIZED instead of VOCALIZED) for users who wish to work with those forms.

The trees are supplied in three forms:

1) with tree tokens (leaves) arising from the VOCALIZED field, for which the excerpt here is:

```
(NP (NOUN+CASE_DEF_GEN kutub+i-)
    (NP (POSS_PRON_3MS -hi)))
```

2) with tree tokens arising from the UNVOCALIZED field:

```
(NP (NOUN+CASE_DEF_GEN ktb)
    (NP (POSS_PRON_3MS h)))
```

3) and with tree tokens arising from the index value:

```
(NP W13
    (NP W14))
```

with the intent that this will make it easier to use any of the variants of the tree token forms.

Despite the presence of these alternate forms of the tree tokens in the release, during the annotation process it is the vocalized tree tokens (i.e., VOCALIZED in the above "treebank-level" example listings), arising from the partition of the SAMA analysis, that are the basis of the tree annotation. The UNVOCALIZED form is a derivative byproduct of this analysis, produced at the end of annotation. Likewise, the INPUT_STRING for the tree token is a derivative byproduct of the full annotation process (although the INPUT_STRING for the source token is not, and is the actual beginning of the annotation process). We discuss in Section 4 the exact procedure used for generating the UNVOCALIZED and INPUT_STRING forms of the tree tokens.

The definition of these derivative fields is in fact not a trivial matter. While the INPUT_STRING for the source token has a perfectly coherent definition (just the original whitespace-delimited token in the source text), the INPUT_STRING for the tree token is not so intuitive, although in the above it seems obvious. Likewise, the UNVOCALIZED value has had an inconsistent definition in the past. We have recently modified the definition and creation of these forms to provide more consistent and meaningful data for users wishing to use these forms, as described in Section 4.

Another issue that arises in the annotation workflow described so far is that it relies upon a SAMA solution being available. This is not always the case, and we have recently modified the annotation pipeline to better handle this situation. This is described in Section 3. (See also Maamouri et al. (2010) for some discussion of the wider context of this annotation pipeline.)

3. Status of Integration with SAMA

As described in Section 1, a significant change in recent releases of the ATB is that the data includes information making explicit the relation between each source token and the morphological analysis (the selected SAMA solution).

Each source token now includes a line for "STATUS," which has one of the values 1-4. We illustrate the meaning of these values with examples from the corpus ATB3-v3.2[5]. All of the examples in this section are from the "pos-level" ("before-treebank") file listing the tokens – that is, the source token listing, because these are the annotation objects that directly relate to SAMA (as opposed to the tree tokens).

Status #1. INCLUDED IN SAMA: The source token and associated solution exactly match one of the possible solutions for this source token in SAMA. That is, the Part-of-Speech (POS), vocalization, and lemma values for the source token exactly match one of the solutions in SAMA for that source token.

For example, this solution:

[5] As of this writing, ATB3-v3.2 is scheduled for publication in April 2010, LDC Catalog Number: LDC2010T08.

```
INPUT_STRING: جندياً
    IS_TRANS: jndyAF
       INDEX: P1W2
     OFFSETS: 4-11
      TOKENS: P1W2-P1W2
      STATUS: 1
       LEMMA: [junodiy~_1]
  UNSPLITVOC: (junodiy~AF)
         POS: NOUN+CASE_INDEF_ACC
         VOC: junodiy~+AF
       GLOSS: soldier + [acc.indef.]
```

has status #1, indicating that the given solution (POS,VOC,GLOSS,LEMMA,UNSPLITVOC) exactly matches one of the SAMA solutions for the input word jndyAF.

Status #2. LIMITED SOLUTION: The given solution is not a possible SAMA solution for the input string, and so has been entered manually as a separate step in the annotation pipeline. The entered solution is of a very limited format, in which there has been no attempt to add the vocalization information in a typical SAMA solution.

For example, this solution:

```
INPUT_STRING: ســـنترال
    IS_TRANS: sntrAl
       INDEX: P4W42
     OFFSETS: 229-236
      TOKENS: P4W49-P4W49
      STATUS: 2
       LEMMA: [TBupdate]
  UNSPLITVOC: None
         POS: FOREIGN
         VOC: sntrAl
       GLOSS: nogloss
```

is a status #2 solution, since the given fields of the solution are not a SAMA solution – the VOC is the same as the INPUT_STRING field, with no additional information entered.

The intent is that status #2 will be reserved for those words that are Arabic but are not expected to have a solution in SAMA, such as TYPO, FOREIGN, etc. This also includes DIALECT words, which are by intent not included in SAMA, which is focused on Modern Standard Arabic. Following is one such example:

```
INPUT_STRING: بتقـــوم
    IS_TRANS: btqwm
       INDEX: P15W7
     OFFSETS: 36-42
      TOKENS: P15W8-P15W8
      STATUS: 2
       LEMMA: None
  UNSPLITVOC: None
         POS: DIALECT
         VOC: btqwm
       GLOSS: nogloss
```

Status #3. PENDING SAMA SOLUTION: This is similar to status #2 in that the given solution is not a SAMA solution for the input string, and so has been entered in an alternate way. However, in this case the solution does have vocalization and other characteristics of a SAMA solution, and so is considered a “pending” SAMA solution. The intent is that these solutions will be subject to further review and eventual inclusion in SAMA.

For example, this solution:

```
INPUT_STRING: بانــه
    IS_TRANS: bAnh
       INDEX: P6W15
     OFFSETS: 68-73
      TOKENS: P6W18-P6W20
      STATUS: 3
       LEMMA: [bi>an~a_1]
  UNSPLITVOC: (bi>an~ahu)
         POS: PREP+SUB_CONJ+PRON_3MS
         VOC: bi+>an~a+hu
       GLOSS: by/with+that+it/he
```

has status #3 because the given solution is not actually a solution included in SAMA 3.1, although it has the complete form of such a solution. (This particular instance appears to an example of the well-known “missing hamza” problem discussed in Buckwalter (2004).)

Status #4. EXCLUDED FROM CHECK WITH SAMA: The source token is a case of punctuation or some other token that by intent is not included in SAMA and therefore excluded from this analysis.

For example, this solution:

```
INPUT_STRING: 650
    IS_TRANS: 650
       INDEX: P1W1
     OFFSETS: 0-4
      TOKENS: P1W1-P1W1
      STATUS: 4
       LEMMA: [DEFAULT]
  UNSPLITVOC: (650)
         POS: NOUN_NUM
         VOC: 650
       GLOSS: nogloss
```

has status #4 because digit numbers are not checked for inclusion in SAMA.

In the ATB3-v3.2 release there are 339,710 source tokens, which are categorized with the following statuses:

Status #1	INCLUDED IN SAMA	287,282
Status #2	LIMITED SOLUTION	939
Status #3	PENDING SAMA SOLUTION	4323
Status #4	EXCLUDED FROM CHECK WITH SAMA	47,156
TOTAL		339,710

Table 1: Categorization of SAMA status of source tokens in ATB3-v3.2.

4. New Algorithm for Creation of INPUT STRING and UNVOCALIZED Tokens

We now return to the issue discussed in Section 2, regarding the creation of the INPUT STRING and UNVOCALIZED tokens at the end of the annotation process.

4.1 INPUT STRING

The INPUT_STRING value for the tree token is a substring of the INPUT_STRING for the source token such that it corresponds in a mostly natural way to the VOCALIZED field for that tree token. For example, in Section 2 the source token with INPUT_STRING “ktbh” yielded two tree tokens with INPUT_STRING values “ktb” and “h”, corresponding to the VOCALIZED values “kutubi” and “hi”.

To take a somewhat less obvious example, the source token “EmA” can receive the following analysis:

```
INPUT_STRING: عما
    IS_TRANS: EmA
       INDEX: P13W13
     OFFSETS: 72-76
      TOKENS: P13W15-P13W16
      STATUS: 1
       LEMMA: [Eam~A_1]
  UNSPLITVOC: (Eam~A)
         POS: PREP+REL_PRON
         VOC: Ean+mA
       GLOSS: from/about/of+what
```

In this case the vocalized solution is Ean+mA, and based on the part-of-speech tags results in two tree tokens, for Ean/PREP and mA/REL_PRON:

```
INPUT_STRING: ع
    IS_TRANS: E
     COMMENT: [Separated]
       INDEX: P13W15
     OFFSETS: 72,73
 UNVOCALIZED: En
   VOCALIZED: Ean-
         POS: PREP
       GLOSS: from/about/of

INPUT_STRING: ما
    IS_TRANS: mA
     COMMENT: []
       INDEX: P13W16
     OFFSETS: 73,76
 UNVOCALIZED: mA
   VOCALIZED: -mA
         POS: REL_PRON
       GLOSS: what
```

This is an example where the “normalization” introduced in SAMA inserts the “n” of “Ean” that is dropped when it appears in the source token text “EmA”. However, the creation of the INPUT_STRING values for the two tree tokens is a partition of the source token text, so “EmA” must be split over the two tree tokens, and here the “mA” in the source token INPUT_STRING belongs with the second tree token, and only the “E” remains for the first tree token.

The algorithm used in earlier data releases to create the INPUT_STRING tokens for the tree tokens sometimes created inappropriate INPUT_STRING tokens. For example, here it would have created the two INPUT_STRING values “Em” and “A”.

To correct this, we have completely revised the procedure for creating these INPUT_STRING tree tokens. The new algorithm can be viewed as a function that takes as input the source token text and vocalized tree tokens, and outputs the source token text in the appropriate way. This requires accounting for the possible types of normalization that might occur in the vocalized tree tokens as a result of normalization included in SAMA, such as the "n" insertion in the above example. This is what we have tried to do, by essentially stepping through the source token and vocalized tree tokens in parallel, and deciding on the proper partition points in the source token text, allowing for discrepancies in characters between the vocalized tree tokens and the source token text that arise from the SAMA normalization.

4.2 UNVOCALIZED

The UNVOCALIZED form of the tree token had a sort of a hybrid definition in earlier releases. For tree tokens that did not result from split source tokens (i.e., the source token resulted in only one tree token), the UNVOCALIZED form was identical to the INPUT_STRING. For source tokens that were split, each UNVOCALIZED form of each resulting tree token was set to be simply the VOCALIZED form with diacritics removed. This hybrid nature of the UNVOCALIZED form is discussed further in Maamouri, Kulick & Bies (2008). The previous UNVOCALIZED form has also been used for parsing (Bikel, 2004; Kulick et al., 2006).

This hybrid definition of the UNVOCALIZED tokens led to inconsistencies in the Treebank, such that instances of tree tokens with the same INPUT_STRING and VOCALIZATION appeared with different UNVOCALIZED strings, as discussed in Maamouri, Kulick & Bies (2008). While the vocalized tree tokens have a clear definition as part of the annotation process, and the INPUT STRING tree tokens also have a reasonably clear meaning (even if nontrivial to obtain), this was not true of the UNVOCALIZED tokens in earlier releases.

We have now simplified the definition to make the UNVOCALIZED tree tokens be the VOCALIZED tree tokens with diacritics stripped out (i.e., treating all tokens in the same way as split tokens were treated previously). We illustrate this change with one example showing the effect of the recent changes on the UNVOCALIZED form.

Under the old method for creation of the UNVOCALIZED form, the source token "Ant$Arh" انتشـــاره (at least under one particular analysis), yielded the following two tree tokens:

```
  VOCALIZED: {inoti$Ar+u-
   IS_TRANS: Ant$Ar
UNVOCALIZED: {nt$Ar

  VOCALIZED: -hu
   IS_TRANS: h
UNVOCALIZED: h
```

with the "h" being a pronominal clitic. The vocalized solution for the first tree token is "{inoti$Ar+u". Since the source token was split, the UNVOCALIZED string for the first was set to the VOCALIZED token with diacritics removed, that is just "{nt$Ar".

However, the source token "Ant$Ar" انتشـــار, again under the old method, yielded one tree token:

```
  VOCALIZED: {inoti$Ar+u
   IS_TRANS: Ant$Ar
UNVOCALIZED: Ant$Ar
```

with the vocalization "{inoti$Ar+u". Since in this case the source token was not split, the UNVOCALIZED token was set simply to the original source text, "Ant$Ar".

Therefore, under the old algorithm there were two tree tokens in these cases, both with the same INPUT_STRING (Ant$Ar) and the same VOCALIZED form ({inoti$Ar+u), but different UNVOCALIZED forms ({nt$Ar and Ant$Ar), with the difference occurring only because in one case the original source token happened to have a pronominal clitic.

Using the new algorithm, the UNVOCALIZED string for both is "{nt$Ar", derived for both instances by removing the short vowels from the same VOCALIZED form, "{inoti$Ar+u". Tokens with the same VOCALIZED form will have the same UNVOCALIZED form, regardless of the orthogonal issue of whether the source token included a clitic or not, because the new algorithm derives all UNVOCALIZED forms from the corresponding VOCALIZED forms.

5. Conclusion

The Arabic Treebank closely integrates SAMA into the annotation workflow, in which the morphological analysis for a token in the Treebank is selected from among the SAMA alternative solutions for that token. However, this integration with SAMA gives rise to various challenges for the annotation workflow and for maintaining the link between the Treebank and SAMA.

In this paper we have discussed how we have overcome these problems with consistent and more precise categorization of each Treebank token for its relationship with SAMA. We also discussed how we have improved the creation of alternative forms of the tokens used in the treebank structures.

As a result of this work, the Arabic Treebank and SAMA can now be viewed as a more tightly integrated unit. This provides valuable data for machine learning experiments on the problem of relating a source token to the correct morphological analysis, in this case mediated through the list of possible SAMA solutions. It also provides a resource relating the different forms of the same underlying token with varying degrees of vocalization, both in terms of how they relate to each other and how they relate to the syntactic structure.

6. Acknowledgements

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.

We would also like to thank David Graff and Fatma Kaddeche for valuable discussions, and the Arabic Treebank annotators for their many contributions.

7. References

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

Daniel Bikel. (2004). On the Parameter Space of Lexicalized Statistical Parsing Models. Dissertation, Department of Computer and Information Sciences, University of Pennsylvania. 2004.

Tim Buckwalter. (2004). Issues in Arabic orthography and morphology analysis. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages* (ACL Semitic Workshop 2004).

Seth Kulick, Ryan Gabbard and Mitchell Marcus. (2006). Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the 5th International Conference on Treebanks and Linguistic* (TLT 2006).

Mohamed Maamouri, Ann Bies and Seth Kulick. (2009). Upgrading and enhancing the Penn Arabic Treebank: A GALE challenge. In Joseph Olive (Ed.), *In progress for publication (book describing work in GALE program).*

Mohamed Maamouri, Ann Bies, Seth Kulick, Wajdi Zaghouani, David Graff and Michael Ciul. (2010). From Speech to Trees: Applying Treebank Annotation to Arabic Broadcast News. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation* (LREC 2010).

Mohamed Maamouri, Seth Kulick and Ann Bies. (2008). Diacritic Annotation in the Arabic Treebank and its Impact on Parser Evaluation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation* (LREC 2008).