Rapid Development of Morphological Analyzers for Typologically Diverse Languages

Seth Kulick and Ann Bies
Linguistic Data Consortium
University of Pennsylvania
3600 Market Street, Suite 810
Philadelphia, PA 19104
{skulick,bies}@ldc.upenn.edu

Abstract

The Low Resource Language research conducted under DARPA's Broad Operational Language Translation (BOLT) program required the rapid creation of text corpora of typologically diverse languages (Turkish, Hausa, and Uzbek) which were annotated with morphological information, along with other types of annotation. Since the output of morphological analyzers is a significant aid to morphological annotation, we developed a morphological analyzer for each language in order to support the annotation task, and also as a deliverable by itself. Our framework for analyzer creation results in tables similar to those used in the successful SAMA analyzer for Arabic (Maamouri et al., 2010), but with a more abstract linguistic level, from which the tables are derived. A lexicon was developed from available resources for integration with the analyzer, and given the speed of development and uncertain coverage of the lexicon, we assumed that the analyzer would necessarily be lacking in some coverage for the project annotation. Our analyzer framework was therefore focused on rapid implementation of the key structures of the language, together with accepting "wildcard" solutions as possible analyses for a word with an unknown stem, building upon our similar experiences with morphological annotation with Modern Standard Arabic and Egyptian Arabic.

Keywords: Morphology, Lexicon, Morphological analyzer

1. Introduction

The Low Resource Language research conducted under DARPA's Broad Operational Language Translation (BOLT) program required the rapid creation of text corpora of typologically diverse languages (Turkish, Hausa, and Uzbek) which were annotated with morphological information, along with other types of annotation. Since the output of morphological analyzers is a significant aid to morphological annotation, we developed a morphological analyzer for each language in order to support the annotation task, and also as a deliverable by itself. We developed a framework for creating an analyzer which results in tables similar to those used in the successful SAMA analyzer for Arabic (Maamouri et al., 2010), but with a more abstract linguistic level, from which the tables are derived. A lexicon was developed from available resources for integration with the analyzer, and given the speed of development and uncertain coverage of the lexicon, we assumed that the analyzer would necessarily be lacking in some coverage for project annotation. Our analyzer framework was therefore focused on rapid implementation of the key structures of the language, together with accepting "wildcard" solutions as possible analyses for a word with an unknown stem, building upon our experience with morphological annotation with Modern Standard Arabic and Egyptian Arabic.

In Section 2. we give an overview of the three interacting aspects of the analyzer infrastructure. In Section 3. we discuss the framework for specifying the possible analyses for a language, and how it interacts with the separate lexicon development. In Section 4. we discuss further the comparison with SAMA, and in Section 5. some additional details of the lexicon/analyzer interaction. Section 6. discusses results and conclusions.

2. Overview

The infrastructure has three interacting aspects:

(a) Specification of Possible Analyses: We developed a framework for the creation of the specification of possible affix combinations for each class of stems in a language, in which the classes are defined by various morphological features. This is where the morphological analysis of the language was done. We describe this framework in more detail in Section 3.

(b) Lexicon: The lexicon LDC created for each language was essentially a dictionary with some part-of-speech, morphological, and gloss information for each entry.

(c) Integration of the analyzer infrastructure: Since the specification of possible affixes in (a) was based on a classification of morphological properties of stems in (b), they were combined to produce a representation of the analyzer information by a combination of determining the morphological class of each entry in the lexicon (b) and putting that in the correct "slot" in the morphological specification (a).

This representation resulting from the integration in (c) was then used by a morphological annotation tool to produce possible analyses for each word, as part of the annotation task in which the possible analyses were presented to an annotator. In the annotation task, the human annotator chose either the correct solution from the possible analyses or a "wildcard" solution (discussed below). The morphological specification used by the annotation tool is purely concatenative, represented in a human-readable text file with a list of possible combinations of affixes that can precede

or follow a stem, in effect compiling out the different orthographic changes that can occur in tokens as a result of certain types of affixation.

The analyzer infrastructure was focused on rapid implementation of the key structures of the language, in particular for accepting "out of vocabulary" (OOV) open-class items based on their morphological properties. We assumed that the analyzer would necessarily be lacking in coverage, given the speed of development, and the uncertain status of available open lexicon material. This was one motivation for the separate development of the lexicon and the specification of possible suffixes in the analyzer.

Also, the interaction between the lexicon and the morphological specification could be, and indeed was, different for each language, depending on the morphological characteristics of the language and the available resources. For example, since Hausa has far less suffixation than Turkish, for Hausa the lexicon held multiple entries for a stem containing the effects of the derivational morphology, with much less information in the morphological specification than for Turkish.

While morphological analyzer development tools such as XFST (Beesley and Karttunen, 2003) or Foma (Hulden, 2009) can be extremely powerful and very useful for grammar organization, they can also be quite fragile and complex when dealing with various phenomena such as long-distance dependencies between morphemes, so they could not be used exclusively for this project. We therefore adopted a hybrid approach, described in Section 3.

3. Development of the Morphological Analysis Specification

The starting point of the morphological specification was the information in the more general grammatical sketch that was developed for each language as part of the project (Strassel and Tracey, 2016). The grammatical sketch is simply a description of the main characteristics of a given language - morphology, syntax, etc., similar to the information that could be found in any overview of a language. It was written from the perspective of being used by a NLP system developer, not by a linguist.

3.1. Abstract Morpheme Organization

While we do not use Foma (Hulden, 2009) for the full analyzer specification, as mentioned above, we do utilize it as a convenient way to implement the specification of the possible sequences of abstract morphemes as described in the grammatical sketch, supplemented by language-specific grammar books where needed. By "possible sequences of abstract morphemes" we mean the possible sequences of morphemes, without accounting for the concrete surface realization of morphemes in particular sequences, since that would depend on particular properties of the stem and other morphemes.

We will use Turkish as a running example, since that is the language that this system was first developed for. In Turkish, two of the main types of vocal harmony with the stem or previous morpheme can be represented with an *E* (realized in the surface form as as *e* or *a*) and an *I* (appearing as *i*, *ı*, *u*, *ü*).

```
LEXICON NOUN_STEM
...
NOUN_STEM_%02/NOUN:x    STATE_NOUN ;
....
NOUN_STEM_%05/NOUN:x    STATE_NOUN ;
....
NOUN_STEM_%08/NOUN:x    STATE_NOUN ;
....
NOUN_STEM_%11/NOUN:x    STATE_NOUN ;
....

LEXICON STATE_NOUN
+STATE-NOUN:x   NOUN_HYPOCORISTIC ;
+STATE-NOUN:x   NOUN_PLURAL ;

LEXICON NOUN_HYPOCORISTIC
+CIk/DIM:x      # ;

LEXICON NOUN_PLURAL
                  NOUN_POSS_FROM_STEM ;
+lEr/PLURAL:x     NOUN_POSS_FROM_PLURAL ;

LEXICON NOUN_POSS_FROM_STEM
                    NOUN_CASE ;
+(I)m/POSS_1S       NOUN_CASE ;
...
LEXICON NOUN_POSS_FROM_PLURAL
                    NOUN_CASE ;
+(I)m/POSS_1S       NOUN_CASE ;
...
```

Figure 1: Foma specification for sequences of abstract morphemes in Turkish

Figure 1 shows an excerpt from the Foma specification. Each of the entries in the `NOUN_STEM` lexicon represents one of the possible morphological classes, as will be discussed in more detail in the following subsections. Each stem class can be followed by the hypocoristic (diminutive) suffix `CIk`, or the plurals `lEr`, as well as the usual possessive endings such as `(I)m`. The parentheses around `I` indicate that the `I` is present or not depending on whether the previous morpheme ends in a consonant or vowel. This is just a mnemonic for naming the suffix, and the actual implementation of this optional `I` is described in Section 3.2. The main point here is that once this information is encoded into the Foma representation, all such combinations can be compiled out into sequences of all possible abstract morphemes, as illustrated in Section 3.3.

3.2. Concrete Morpheme Specification

In Section 3.1. we described the specification of the possible sequences of the abstract morphemes. In this section we describe the specifications of how those abstract morphemes get realized as concrete morphemes respecting vowel harmony and other aspects of morphological change. The basic idea is that for each abstract morpheme we list all possible realizations of that morpheme, and for each realization we list the properties of the preceding morpheme (the "input" morpheme) that determine that that is the correct realization. We also list, for each surface morpheme, what the "output" properties are for that morpheme. The

```
sdefn:NOUN_STEM_02/NOUN nstem_02  V:FR C:VL
sdefn:NOUN_STEM_05/NOUN nstem_05  V:FU C:VL
sdefn:NOUN_STEM_08/NOUN nstem_08  V:BR C:VL
sdefn:NOUN_STEM_11/NOUN nstem_11  V:BU C:VL

defn:CIk/DIM
! ends in VL c -> ç
cük     V:FR C:OTHER,NONE       V:FR C:VL
cik     V:FU C:OTHER,NONE       V:FU C:VL
cuk     V:BR C:OTHER,NONE       V:BR C:VL
cık     V:BU C:OTHER,NONE       V:BU C:VL
çük     V:FR C:VL       V:FR C:VL
çik     V:FU C:VL       V:FU C:VL
çuk     V:BR C:VL       V:BR C:VL
çık     V:BU C:VL       V:BU C:VL

defn:lEr/PLURAL
ler     V:FR,FU C:VL,OTHER,NONE V:FU C:OTHER
lar     V:BR,BU C:VL,OTHER,NONE V:BU C:OTHER

defn:lErI/PLURAL&POSS_3P
lerü    V:FR C:r,l,VL,OTHER,NONE        V:FR C:NONE
leri    V:FU C:r,l,VL,OTHER,NONE        V:FU C:NONE
laru    V:BR C:r,l,VL,OTHER,NONE        V:BR C:NONE
ları    V:BU C:r,l,VL,OTHER,NONE        V:BU C:NONE

defn:(I)m/POSS_1S
! ends in vowel -> m
! ends in consonant -> Im
m       V:FR C:NONE     V:FR C:OTHER
m       V:FU C:NONE     V:FU C:OTHER
m       V:BR C:NONE     V:BR C:OTHER
m       V:BU C:NONE     V:BU C:OTHER
üm      V:FR C:VL,OTHER V:FR C:OTHER
im      V:FU C:VL,OTHER V:FU C:OTHER
um      V:BR C:VL,OTHER V:BR C:OTHER
ım      V:BU C:VL,OTHER V:BU C:OTHER
```

Figure 2: Specification for concrete realizations of morphemes in Turkish

relevant properties depend on the specific characteristics of the language, and can be adjusted depending on the language.

For example, for Turkish the important properties of the preceding morpheme are the frontedness and roundedness of the final vowel, whether the ending consonant is voiceless, and so on. Specifically, we choose to specify a list of categories for the last vowel, which can be one of `FR/FU/BR/BU`, where `F/B` stands for front/back and `R/U` for round/unround. This is implemented as a set of values for the `V` key in a (hash) map. Similarly, we have a `C` key in the map, with the possible values for the final consonant, which can be `VL` (voiceless), `NONE` (no final consonant), `OTHER` (voiced consonant). The relevant properties are of course different for other languages, and for Hausa we use the properties of number and gender as the critical information for determining the surface forms.

Each realization of an abstract morpheme has the format

```
form input-properties output-properties
```

where the `input-properties` and `output-properties` are maps with values for `V` and `C`.

Some examples are shown in Figure 2. The definition for the abstract morpheme `lEr` for the `PLURAL` can appear as either *ler* or *lar*. The former is the case if the `V` value for the preceding morpheme is `FR` or `FU` (the `C` value is irrelevant, so all are listed as possibilities), and the latter is the case if the `V` value for the preceding morpheme is `BR` or `BU` (and again the `C` value is irrelevant). If *ler*, the output property for `V` is `FU` (since *e* is front/unrounded), and if *lar* it is `BU` (since *a* is back/unrounded). In both cases the final consonant is not voiceless, so the `C` value is `OTHER`.

The abstract morpheme `(I)m/POSS_1S` has a slightly different twist, in which the `I` appears or not (and in different realizations) depending on whether the previous morpheme ends in a consonant. This is accomplished by simply listing all the possibilities. Note that in the final four *üm, im, um, ım* the `I` appears because the input `C` value is not `NONE`, and also the exact form that the `I` takes is dependent on the front/roundedness of the preceding morpheme. For the first four cases without the `I`, the output vowel property (`FR`, etc.) is just passed along from the previous morpheme.

The properties used for specifying the surface forms of abstract morphemes in Hausa are of course different than for Turkish. For Hausa, we use properties of number and of

```
defn:n/DEFINITE
n  G:M N:S        G:M N:S V:ANY
n  G:M,F N:P      G:M N:S V:ANY
r  G:F N:S        G:F N:S V:ANY
```

Figure 3: Specification for concrete realizations of the definite suffix in Hausa

gender as the critical information for determining the surface forms. In Figure 3 we show a small example of this, in which the definite suffix appears as *r* if the the preceding morpheme is feminine and singular, and *n* otherwise.

3.2.1. Stem Placeholders

The entries under NOUN_STEM in Figure 2 illustrate how we use various stem names as placeholders for stems with different morphological properties. For example, NOUN_STEM_02 stands for all noun stems in which the last vowel is front and rounded (V:FR) and the ending consonant is voiceless (C:VL), etc. There are not separate input and output properties for the stems, since for Turkish they are the starting point of the sequences, and so they have only output properties. In Section 3.5. we discuss how this is linked to the lexicon, which is collected separately.

3.3. Generation of Abstract and Concrete Morpheme Sequences

With the specifications of the abstract morpheme sequences as described in Section 3.1. and the concrete morpheme realizations as described in Section 3.2., we can combine the two to create the suffix sequences for a language.

It is convenient to use the Foma processing to create all the possibilities of abstract morpheme sequences, as shown in Figure 4. The four different NOUN_STEM entries shown in Figure 1 each get their own set of possible abstract suffix sequences, which are all identical because each NOUN_STEM goes to the state NOUN_STEM. The reason for this is that they work together with the stem names as placeholders in the concrete abstract specification, as discussed in Section 3.2.1. That is, the line

```
NOUN_STEM_02/NOUN+STATE-NOUN+lEr/PLURAL
```

means "a noun with morphological properties V:FR C:VL followed by the plural morpheme lEr", and the line

```
NOUN_STEM_05/NOUN+STATE-NOUN+lEr/PLURAL
```

means "a noun with morphological properties V:FU C:VL followed by the plural morpheme lEr".

We then use some fairly simple code to run through each such abstract sequence, making each morpheme concrete as specified in Figure 5, and updating the morphological properties to be used for each morpheme using the "output" morphological properties for each morpheme in the same table as the "input" properties for the next morpheme.

The result is that the abstract morpheme sequences in Figure 4 are now instantiated as in Figure 5.[1]

[1]One of the benefits of the flexible way in which properties are passed along the morphemes is that longer-distance morphological properties can be handled with relative ease, which might not be so easily done in more strictly specified finite-state systems.

3.4. Reorganization by Suffix Sequences

The description so far shows how we have generated the set of possible suffix sequences, in effect compiling out the set of possible sequences, keyed by the morphological class of the stem. The next step in the processing is to rearrange the information so that it is keyed by suffix sequence, mapping to the possible stems that can take that suffix sequence. For example, in the excerpts shown in Figure 5, *ler* can be a plural suffix for both noun stem class two and five.

3.5. Synchronization with Lexicon

LDC created lexicons for each language as a separate part of the project (Strassel and Tracey, 2016). The lexicon for a language was collected from some available source depending on what is available for the particular language, with each word in the lexicon needing to be integrated into the analyzer. We did this by examining each stem and categorizing it as one of the possible stem classes for the given part-of-speech.

For example, for Turkish there would be categorizations based on the frontedness and roundedness of the final vowel, whether the ending consonant is voiceless, and other categories that affect the realization of morphemes. The classifications for Hausa are far simpler, since we did not attempt to encode the tonal system, which is rarely written in the orthography and was not part of the annotation task. For Hausa, the classifications of the open class words are mostly with regard to the number and gender, which e.g. affect the form of the construct and definite suffixes.

For Turkish, one additional step had to be taken due to our separation of the lexicon and analyzer information as two separate components that are then joined together. In Turkish, the stem itself can undergo alternations based on surrounding suffixes. For example, the stem *kitap* (book) changes to *kitab* when followed by the suffix *ı* (accusative case). To handle this, the stem categorization code in the analyzer also automatically creates alternate forms, which then also receive their own classification.

4. Comparison with SAMA and Use of Wildcards

The end result of the process described in Section 3. is an analyzer that is organized much like the successful SAMA analyzer for Modern Standard Arabic (Maamouri et al., 2010). In SAMA, the morphemes are grouped into three tables with morpheme sequences for the prefixes, suffixes, and the stems, the categories for the stems mediating which affixes and stems can go together. The analysis then consists of partitioning a word into all possible subsequences, and looking up the sequences in hash tables for the stems and sequences.[2]

Our analyzer as described above can be implemented in the same way, in addition to being implemented in the annotation tool mentioned in the introduction. A given word is

[2]There are also two tables in SAMA specifying which prefixes and suffixes are compatible, which we have not needed to implement for this work yet. The description for Turkish is only for suffix sequences, but could be extended to include prefix sequences for other languages without much trouble.

```
NOUN_STEM_02/NOUN+STATE-NOUN+CIk/DIM
NOUN_STEM_02/NOUN+STATE-NOUN+lEr/PLURAL
NOUN_STEM_02/NOUN+STATE-NOUN+(I)m/POSS_1S
NOUN_STEM_02/NOUN+STATE-NOUN+lEr/PLURAL+(I)m/POSS_1S

NOUN_STEM_05/NOUN+STATE-NOUN+CIk/DIM
NOUN_STEM_05/NOUN+STATE-NOUN+lEr/PLURAL
NOUN_STEM_05/NOUN+STATE-NOUN+(I)m/POSS_1S
NOUN_STEM_05/NOUN+STATE-NOUN+lEr/PLURAL+(I)m/POSS_1S

NOUN_STEM_08/NOUN+STATE-NOUN+CIk/DIM
NOUN_STEM_08/NOUN+STATE-NOUN+lEr/PLURAL
NOUN_STEM_08/NOUN+STATE-NOUN+(I)m/POSS_1S
NOUN_STEM_08/NOUN+STATE-NOUN+lEr/PLURAL+(I)m/POSS_1S

NOUN_STEM_11/NOUN+STATE-NOUN+CIk/DIM
NOUN_STEM_11/NOUN+STATE-NOUN+lEr/PLURAL
NOUN_STEM_11/NOUN+STATE-NOUN+(I)m/POSS_1S
NOUN_STEM_11/NOUN+STATE-NOUN+lEr/PLURAL+(I)m/POSS_1S
```

Figure 4: Abstract morpheme sequences generated by the specification in Figure 1.

```
nstem_02/NOUN+çük/DIM
nstem_02/NOUN+ler/PLURAL
nstem_02/NOUN+üm/POSS_1S
nstem_02/NOUN+ler/PLURAL+im/POSS_1S

nstem_05/NOUN+çik/DIM
nstem_05/NOUN+ler/PLURAL
nstem_05/NOUN+im/POSS_1S
nstem_05/NOUN+ler/PLURAL+im/POSS_1S

nstem_08/NOUN+çuk/DIM
nstem_08/NOUN+lar/PLURAL
nstem_08/NOUN+um/POSS_1S
nstem_08/NOUN+lar/PLURAL+ım/POSS_1S

nstem_11/NOUN+çık/DIM
nstem_11/NOUN+lar/PLURAL
nstem_11/NOUN+ım/POSS_1S
nstem_11/NOUN+lar/PLURAL+ım/POSS_1S
```

Figure 5: Concrete morpheme sequences corresponding to the abstract sequences in Figure 4, generated using the specification in Figure 2.

split into every possible partition of a stem and suffix sequence, and the potential suffix sequence is looked up in a hash table. If it is in the hash table, it lists the possible noun stems that can take that suffix. Wintner (2008) takes a similar approach.

For example, if the word is *milletler* (nations), the analyzer partitions the word into possible subsequences, such as *mil+letler*, *milletl+er*, etc. Of the possibilities, only *millet+ler* will have a result when the suffix sequence *ler* is looked up, showing that it takes various noun stem classes, including noun stems two and five, as discussed in Section 3.4. A separate list is maintained of what stems belong to each class, as discussed in Section 3.5., and with *millet* in noun class five, it returns the analysis `millet/NOUN+ler/PLURAL`.[3]

While the end-product is very similar to the SAMA organization, it has the important difference that there is a more abstract linguistic level, from which the compiled out version is derived. This more abstract linguistic level is the specification described in Section 3.1. and Section 3.2. This more abstract level makes it easier to maintain the linguistic aspects of the analyzer.

We also maintain a format for specifying the morphological properties of the stem classes based on flexible key/value entries in a map. This makes it more convenient than in SAMA to classify the possible stems for a suffix, including in terms of the morphological properties that would be needed by a stem to take a particular suffix sequence. We used this property in order to handle OOV items, on the assumption that whatever lexicon we used would necessarily be lacking in coverage for the project annotation, even if the suffix sequences could be made close to complete.

This framework builds upon our experience with development of Arabic analyzers starting with the Arabic Treebank (Kulick et al., 2010) and moving to the Egyptian Arabic Treebank (Maamouri et al., 2014), in which work moved from using an extensive already-existing analyzer for Modern Standard Arabic (MSA) to the simultaneous annotation and development of an analyzer for Egyptian Arabic (Eskander et al., 2013). Even as part of the earlier annotation using SAMA for MSA, "wildcard" solutions were used, which provided possible analyses for a word even when it had an unknown stem. For SAMA, however, this was limited to the use of proper names. For Egyptian Arabic Treebank annotation, the use of wildcards in the annotation was greatly increased in order to handle OOV items, as the CALIMA analyzer (Habash et al., 2012; Maamouri et al., 2014) was simultaneously being developed. However, even then this handling was suboptimal because there was no general classification of what suffix combinations could

[3]Another possible analysis is `millet/NOUN+ler/3P`, due to the ambiguity of *ler*, which we have not discussed here.

follow stems with particular morphological properties.
In contrast, the work reported here integrates this annotation need into the development of the morphological specification, instead of being added on after the analyzer development.

5. Lexicon-Analyzer Tradeoffs, and Integration of Existing Resources

LDC produced a lexicon for each of the languages, along with the manual morphological annotation. There was a tight connection between the lexicon and the analyzer framework. As described in Section 3., the latter created all possible affix combinations for stems classified as having various morphological properties. The creation of the text file used with the annotation tool combined this information together with the information in the lexicon about the stems. As mentioned above, each word in the lexicon was analyzed as to its morphological properties, and then treated accordingly in the analyzer.
The interaction between the lexicon and analyzer was somewhat different for Turkish as opposed to Hausa, however. For Turkish, the analyzer was developed using morphological information from the grammatical sketch and grammar books (Göksel and Kerslake, 2011; Ketrez, 2012). This process was mostly independent of the lexicon development, aside from some sample words chosen to represent each morphological class.
For Hausa, the situation was very different. There is very little suffixation in Hausa, especially as compared to Turkish, and much of the derivational and inflectional morphology in Hausa consists of vowel changes, which are non-concatenative. Also, there were open sources of Turkish vocabulary available on the web in an accessible electronic form, but that was not the case for Hausa. Therefore, we relied on various existing reference works for Hausa (Newman, 2000; Newman, 2007; Bargery, 1934). The available dictionary resources were never meant to be machine-readable, and there was a considerable degree of non-standardization in the entries. However, we developed some heuristics to incorporate information from these materials, and the resulting lexicon groups derivational forms together with the stem in many cases. Although there are some standard suffix cases in Hausa that are treated as such in the analyzer (e.g., construct and definite suffixes, and the pronoun suffixes), the Hausa analyzer was far more dependent on the lexicon than was the case for Turkish.
The process for Uzbek was radically different than for Turkish and Hausa. For Uzbek, the task was to convert a pre-existing analyzer from the REFLEX LCTL Uzbek language pack into a form that could be used with the morphological annotation tool for this project. The existing Uzbek analyzer was written in Hunmorph (Trón et al., 2005), and our conversion process was essentially a reverse-engineering process. The final resulting Uzbek analyzer is in the same form as the Turkish and Hausa analyzers.

6. Results and Conclusion

We carried out an iterative process on the analyzer development, testing it over the 10,000 words in each language that were selected for morphological annotation, focusing on the most frequent words in the corpus. For Turkish, the coverage was at 85.8%, meaning that of the 10,000 tokens, 85.8% had a solution within the analyzer. For Hausa, the coverage was 77.1%. In both cases, this leaves aside OOV words that did not have a solution, but for which the suffixes were recognized and a "wildcard" solution proposed. The annotation task involved annotating a variety of text genres, including informal text such as discussion forms and microblogs, which often include non-standard orthography. In both cases, but particularly for Hausa, there were problematic cases caused by the non-standard nature of the text, especially non-standard word tokenization (such as *abinda* for *abin da*, or *dake* for *da ke*).
The approach described here was successful for rapid development of an analyzer suitable for morphological annotation, and future directions could expand this approach to a wider range of languages. There are also a number of other possibilities for improving the approach taken here. For example, while right now the possible morphological classes of the stems are hard-coded based on the morphological properties of a language, they could instead be automatically derived from the range of properties that affect the realization of the morphemes.
The resources described in this paper have been distributed to performers in the DARPA BOLT and LORELEI programs (Linguistic Data Consortium, 2015b; Linguistic Data Consortium, 2016; Linguistic Data Consortium, 2015a), and will be published in the LDC catalog in 2016.

7. Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract Nos. HR0011-11-C-0145 and HR0011-12-C-0014. The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

8. Bibliographical References

Bargery, G. P. (1934). *A Hausa-English Dictionary and English-Hausa Vocabulary*. Oxford University Press.

Beesley, K. and Karttunen, L. (2003). *Finite State Morphology*. CSLI.

Eskander, R., Habash, N., Bies, A., Kulick, S., and Maamouri, M. (2013). Automatic correction and extension of morphological annotations. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics.

Göksel, A. and Kerslake, C. (2011). *Turkish: An Essential Grammar*. Rougledge Essential Grammars.

Habash, N., Eskander, R., and Hawwari, A. (2012). A morphological analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*.

Hulden, M. (2009). Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*.

Ketrez, F. N. (2012). *A Student Grammar of Turkish*. Cambridge University Press.

Kulick, S., Bies, A., and Maamouri, M. (2010). Consistent and flexible integration of morphological annotation in the Arabic Treebank. In *Language Resources and Evaluation (LREC)*.

Maamouri, M., Bies, A., Kulick, S., Ciul, M., Habash, N., and Eskander, R. (2014). Developing an Egyptian Arabic treeebank: Impact of dialectal morphology on annotation and tool development. In *Proceedings of LREC2014: 9th International Conference on Language Resources and Evaluation*.

Newman, P. (2000). *The Hausa Language - An Encyclopedic Reference Grammar*. Yale University Press.

Newman, P. (2007). *A Hausa-English Dictionary*. Yale University Press.

Strassel, S. and Tracey, J. (2016). Lorelei language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of LREC2016: Tenth International Conference on Language Resources and Evaluation*.

Trón, V., Gyepesi, G., Halácsy, P., Kornai, A., Németh, L., and Varga, D. (2005). Hunmorph: open word source analysis. In *Proceedings of ACL SIGLEX Workshop on Software*.

Wintner, S. (2008). Strengths and weaknesses of finite-state technology: a case study in morphological grammar development. *Natural Language Engineering*, 14:457–469.

9. Language Resource References

Linguistic Data Consortium. (2015a). BOLT LRL Hausa representative language pack v1.2. LDC2015E70.

Linguistic Data Consortium. (2015b). BOLT LRL Turkish representative language pack v2.2. Linguistic Data Consortium2014E115.

Linguistic Data Consortium. (2016). BOLT LRL Uzbek representative language pack v1.0. LDC2016E29.

Maamouri, M., Graff, D., Bouziri, B., Krouna, S., Bies, A., and Kulick, S. (2010). Standard Arabic morphological analyzer (SAMA) version 3.1. Linguistic Data Consortium LDC2010L01.