HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya

Michael Gasser
Indiana University
Bloomington, Indiana, USA
gasser@cs.indiana.edu

Abstract

Despite its linguistic complexity, the Horn of Africa region includes several major languages with more than 5 million speakers, some crossing the borders of multiple countries. All of these languages have official status in regions or nations and are crucial for development; yet computational resources for the languages remain limited or non-existent. Since these languages are complex morphologically, software for morphological analysis and generation is a necessary first step toward nearly all other applications. This paper describes a resource for morphological analysis and generation for three of the most important languages in the Horn of Africa, Amharic, Tigrinya, and Oromo.

1 Language in the Horn of Africa

The Horn of Africa consists politically of four modern nations, Ethiopia, Somalia, Eritrea, and Djibouti. As in most of sub-Saharan Africa, the linguistic picture in the region is complex. The great majority of people are speakers of Afro-Asiatic languages belonging to three sub-families: Semitic, Cushitic, and Omotic. Approximately 75% of the population of almost 100 million people are native speakers of four languages: the Cushitic languages Oromo and Somali and the Semitic languages Amharic and Tigrinya. Many others speak one or the other of these languages as second languages. All of these languages have official status at the national or regional level.

All of the languages of the region, especially the Semitic languages, are characterized by relatively complex morphology. For such languages, nearly all forms of language technology depend on the existence of software for analyzing and generating word forms. As with most other sub-Saharan languages, this software has previously not been available. This paper describes a set of Python programs called HornMorpho that address this lack for three of the most important languages, Amharic, Tigrinya, and Oromo.

2 Morphological processingn

2.1 Finite state morphology

Morphological analysis is the segmentation of words into their component morphemes and the assignment of grammatical morphemes to grammatical categories and lexical morphemes to lexemes. **Morphological generation** is the reverse process. Both processes relate a **surface** level to a **lexical** level. The relationship between the levels has traditionally been viewed within linguistics in terms of an ordered series of phonological rules.

Within computational morphology, a very significant advance came with the demonstration that phonological rules could be implemented as **finite state transducers** (Kaplan and Kay, 1994) (FSTs) and that the rule ordering could be dispensed with using FSTs that relate the surface and lexical levels directly (Koskenniemi, 1983), so-called "two-level" morphology. A second important advance was the recognition by Karttunen et al. (1992) that a cascade of composed FSTs could implement the two-level model. This made possible quite complex finite state systems, including ordered **alternation rules** representing context-sensitive variation in the phonological or orthographic shape of morphemes, the **morphotactics** characterizing the possible sequences of morphemes (in canonical form) for a given word class, and a **lexicon**. The key feature of such systems is that, even though the FSTs making up the cascade must be composed in a particular order, the result of composition is a single FST relating surface and lexical levels directly, as in two-level morphology. Because of the invertibility of FSTs, it is a simple matter to convert an analysis FST (surface input

Conference on Human Language Technology for Development, Alexandria, Egypt, 2-5 May 2011.

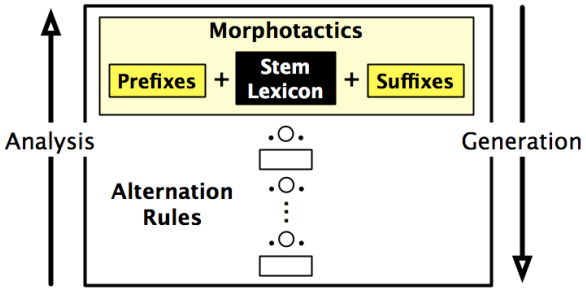

Figure 1: Basic architecture of lexical FSTs for morphological analysis and generation. Each rectangle represents an FST; the outermost rectangle is the full FST that is actually used for processing. ".o." represents composition of FSTs, "+" concatenation of FSTs.

to lexical output) to one that performs generation (lexical input to surface output).

This basic architecture, illustrated in Figure 1, consisting of a cascade of composed FSTs representing (1) alternation rules and (2) morphotactics, including a lexicon of stems or roots, is the basis for the system described in this paper.

We may also want to handle words whose roots or stems are not found in the lexicon, especially when the available set of known roots or stems is limited. In such cases the lexical component is replaced by a phonotactic component characterizing the possible shapes of roots or stems. Such a "guesser" analyzer (Beesley and Karttunen, 2003) analyzes words with unfamiliar roots or stems by positing *possible* roots or stems.

2.2 Semitic morphology

These ideas have revolutionized computational morphology, making languages with complex word structure, such as Finnish and Turkish, far more amenable to analysis by traditional computational techniques. However, finite state morphology is inherently biased to view morphemes as sequences of characters or phones and words as concatenations of morphemes. This presents problems in the case of **non-concatenative morphology**, for example, discontinuous morphemes and the **template morphology** that characterizes Semitic languages such as Amharic and Tigrinya. The stem of a Semitic verb consists of a **root**, essentially a sequence of consonants, and a **template** that inserts other segments between the root consonants and possibly copies certain of the consonants. For example, the Amharic verb root *sbr* 'break' can combine with roughly 50 different templates to form stems in words such as ይሰብራል *yi-sɛbr-al* 'he breaks', ተሰበረ *tɛsɛbbɛr-ɛ* 'it was broken', ላሰብረው *l-assɛbbir-ɛw*, 'let me cause him to break something', ሰባባሪ *sɛbabar-i* 'broken into many pieces'.

A number of different additions to the basic FST framework have been proposed to deal with non-concatenative morphology, all remaining finite state in their complexity. A discussion of the advantages and drawbacks of these different proposals is beyond the scope of this paper. The approach used in our system is one first proposed by Amtrup (2003), based in turn on the well studied formalism of weighted FSTs. In brief, in Amtrup's approach, each of the arcs in a transducer may be "weighted" with a feature structure, that is, a set of grammatical feature-value pairs. As the arcs in an FST are traversed, a set of feature-value pairs is accumulated by unifying the current set with whatever appears on the arcs along the path through the transducer. These feature-value pairs represent a kind of memory for the path that has been traversed but without the power of a stack. Any arc whose feature structure fails to unify with the current set of feature-value pairs cannot be traversed.

The result of traversing such an FST during morphological analysis is not only an output character sequence, representing the root of the word, but a set of feature-value pairs that represents the grammatical structure of the input word. In the generation direction, processing begins with a root and a set of feature-value pairs, representing the desired grammatical structure of the output word, and the output is the surface wordform corresponding to the input root and grammatical structure. In Gasser (2009) we showed how Amtrup's technique can be applied to the analysis and generation of Tigrinya verbs. For an alternate approach to handling the morphotactics of a subset of Amharic verbs, within the context of the Xerox finite state tools (Beesley and Karttunen, 2003), see Amsalu and Demeke (2006).

Although Oromo, a Cushitic language, does not exhibit the root+template morphology that is typical of Semitic languages, it is also convenient to handle its morphology using the same technique because there are some long-distance dependencies and because it is useful to have the grammatical output that this approach yields for analysis.

3 HornMorpho

HornMorpho is a set of Python programs for analyzing and generating words in Amharic, Tigrinya, and Oromo. A user interacts with the programs through the Python interpreter. HornMorpho is available for download, under the GPL3 license, at `http://www.cs.indiana.edu/~gasser/Research/software.html`. Complete documentation is included with the downloaded archive.

For each language, HornMorpho has a lexicon of verb roots and (except for Tigrinya) noun stems.[1] For Amharic, the lexicon is derived from the Amharic-English dictionary of Aklilu (1987), which is available under the Creative Commons Attribution-Noncommercial 3.0 United States License at `http://nlp.amharic.org/resources/lexical/word-lists/dictionaries/`; there are currently 1,851 verb roots and 6,471 noun stems. For Oromo the lexicon of verb and noun roots is extracted from the dictionaries of Gragg (1982) and Bitima (2000); there are currently 4,112 verb roots and 10,659 noun stems. For Tigrinya, the lexicon of verb roots is derived from Efrem Zacarias' (2009) online dictionary, accessible at `http://www.memhr.org/dic/`; there are currently 602 verb roots.

3.1 System architecture

The full morphology processing system (see Figure 1) consists of analysis and generation FSTs for each language. For Amharic and Tigrinya there are separate lexical and "guesser" FSTs for each processing direction. Verbs (all three languages) and nouns (Amharic and Oromo only) are handled by separate FSTs. Amharic and Oromo have separate FSTs for verb segmentation, as opposed to grammatical analysis, and Amharic has a separate FST for orthography-to-phonology conversion.

Each of these FSTs in turn results from the composition of a cascade of simpler FSTs, each responsible for some aspect of morphology. The most complicated cases are Amharic and Tigrinya verbs, which we discuss in more detail in what follows and illustrate in Figure 2. At the most abstract (lexical) end is the heart of the system, the morphotactic FST. Most of the complexity is within the FST responsible for the stem. The stem FST is composed from a sequence of five simpler FSTs representing stem-specific alternation rules and the legal sequences of consonants and vowels making up the stem. For the lexical, but not the guesser, FSTs, a further FST containing the lexicon of known roots is part of the stem cascade.

Prefix and suffix FSTs are concatenated onto the stem FST to create the full verb morphotactic FST. The remaining FSTs (15 of them for Amharic verbs, 17 for Tigrinya verbs) implement alternation rules that apply to the word as a whole, including allomorphic rules and general phonological or orthographic rules.

The figure shows the analysis of the Amharic verb የማትሰደበውም *yəmmatissəddəbəwm* 'who (she) is also not insulted'. The word is input in the Ge'ez orthography used for Amharic and Tigrinya. This writing system fails to indicate consonant gemination as well as the epenthetic vowel *ɨ*, which is introduced to break up some conso-

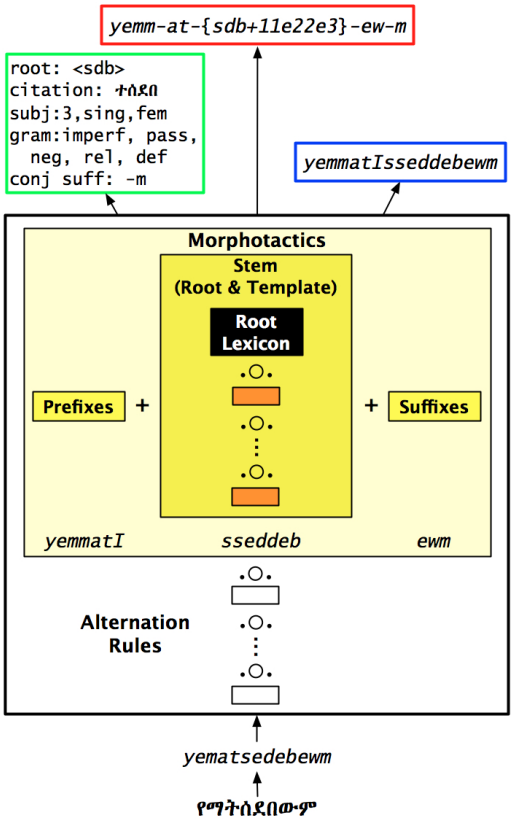

Figure 2: Architecture of Amharic verb analysis FST. Shown: analysis of the verb የማትሰደበውም 'who (she) is also not insulted'. Output analyses: `anal_word` (green border); `seg_word` (red border); `phon_word` (blue border).

[1] Amharic adjectives, which behave much like nouns, are grouped with nouns in the system.

nant clusters in both languages. Gemination is extremely important for natural speech synthesis in Amharic and Tigrinya, so it is crucial to be able to restore it in text-to-speech applications. There is in fact relatively little ambiguity with respect to gemination, but gemination is so tied up with the morphology that a relatively complete morphological analyzer is necessary to perform the restoration. HornMorpho has this capacity.

The word is first romanized to *yematsedebewm*.[2] At this stage, none of the consonants is geminated, and the epenthetic vowel is missing in the romanized form. Processing is then handled by the single analysis FST, but to understand what goes on, it is still to convenient to think of the process in terms of the equivalent cascade of simpler FSTs operating in sequence. The first FST in the cascade performs orthographic-to-phonological conversion, resulting in all possible pronunciations of the input string, including the correct one with the appropriate consonants geminated. This form and the other surviving strings are processed by the intervening phonological FSTs, each responsible for an alternation rule. Among the strings that survive as far as the morphotactic FST is the correct string, *yemmatIsseddebewm*, which is analyzable as the pair of prefixes *yemm-at*, the stem *sseddeb*, and the pair of suffixes *ew-m*.

The stem is processed by the stem FST, which extracts the root, *sdb*, and various grammatical properties, including the fact that this form is passive. The lexical analyzer includes all of the verb roots known to the system within the stem FST, whereas the guesser analyzer includes only information about sequences of consonants making up possible roots in the language. For example, if the consonant sequence *s,d,b* in the original word were replaced by a fictitious root *mbz*, the guesser analyzer (but not the lexical analyzer) would posit this as the root of the word. The final output analyses are shown at the top of Figure 2. The three possibilities correspond to three different HornMorpho functions, discussed in the next section.

3.2 Functions

Each of the functions for morphological analysis has two versions, one for analyzing single words, the other for analyzing all of the words in a file. The functions `anal_word` and `anal_file` take input words and output a root or stem and a grammatical analysis. In Figure 2, the output of `anal_word` is outlined in green. The input word has the root *sdb* 'insult' and the citation form ተሰደበ; has a third person singular feminine subject; is in the imperfective tense/aspect; is relativized, negative, and definite; and has the conjunctive suffix *-m*.

For Amharic and Oromo, there are two additional analysis functions, `seg_word` and `seg_file`, which segment input verbs (also nouns for Oromo) into sequences of morphemes. In the example in Figure 2, the output of `seg_word` is shown in red. The constituent morphemes are separate by hyphens, and the stem is enclosed in brackets. The root and template for the stem are separated by a plus sign. The template notation $11e22e3$ indicates that the first and second root consonants are geminated and the vowel *e* is inserted between the first and second and second and third root consonants.

For Amharic only, there are further functions, `phon_word` and `phon_file`, which convert the input orthographic form to a phonetic form as would be required for text-to-speech applications. In the figure, the output of `phon_word` is outlined in blue. Three of the consonants are geminated, and the epenthetic vowel (romanized as *I*) has been inserted to break up the cluster *tss*.

Below are more examples of the analysis functions, as one would call them from the Python interpreter. Note that all of the HornMorpho functions take a first argument that indicates the language. Note also that when a wordform is ambiguous (Example 3), the analysis functions return all possible analyses.

Example 1 `anal_word` (Tigrinya)

```
>>> anal_word('ti', 'ብዘጋጥመና')
Word:  ብዘጋጥመና
POS:verb, root:<gTm>, cit:ኣጋጠመ
subject:  3, sing, masc
object:  1, plur
grammar:  imperf, recip, trans, rel
preposition:  bI
```

Example 2 `seg_word` (Oromo)

```
>>> seg_word('om', 'dhukkubdi')
dhukkubdi:  dhukkub-t-i
```

There is a single function for generation, `gen`, which takes a stem or root and a set of grammat-

[2] HornMorpho uses an ASCII romanization scheme developed by Firdyiwek and Yaqob (1997).

Example 3 `phon_word` (Amharic)

```
>>> phon_word('am', 'ይመታሉ')
yImetallu yImmettallu
```

ical features. For each part of speech, there is a default set of features, and the features provided in the function call modify these. In order to use `gen`, the user needs to be familiar with the HornMorpho conventions for specifying grammatical features; these are described in the program documentation.

With no grammatical features specified, `gen` returns the canonical form of the root or stem, as in the Oromo example 4 (*sirbe* is the third person singular masculine past form of the verb). Example 5 is another Oromo example, with additional features specified: the subject is feminine, and the tense/mood is present rather than past.

Example 4 `gen` (Oromo 1)

```
>>> gen('om', 'sirb')
sirbe
```

Example 5 `gen` (Oromo 2)

```
>>> gen('om','sirb','[sb=[+fem],tm=prs]')
sirbiti
```

4 Evaluation

Evaluating HornMorpho is painstaking because someone familiar with the languages must carefully check the program's output. A useful resource for evaluating the Amharic and Tigrinya analyzers is the word lists compiled by Biniam Gebremichael's web crawler, available on the Internet at `http://www.cs.ru.nl/~biniam/geez/crawl.php`. The crawler extracted 227,984 unique Tigrinya wordforms and 397,352 unique Amharic wordforms.

To evaluate the Amharic and Tigrinya analyzers in HornMorpho, words were selected randomly from each word list, until 200 Tigrinya verbs, 200 Amharic verbs, and 200 Amharic nouns and adjectives had been chosen. The `anal_word` function was run on these words, and the results were evaluated by a human reader familiar with the languages. An output was considered correct only if it found all legal combinations of roots and grammatical structure for a given wordform and included no incorrect roots or structures. The program made 8 errors on the Tigrinya verbs (96% accuracy), 2 errors on the Amharic verbs (99% accuracy), and 9 errors on the Amharic nouns and adjectives (95.5% accuracy).

To test the morphological generator, the `gen` function was run on known roots belonging to all of the major verb root classes.[3] For each of these classes, the program was asked to generate 10 to 25 verbs depending on the range of forms possible in the class, with randomly selected values for all of the different dimensions, a total of 330 tests. For Amharic, the program succeeded on 100% of the tests; for Tigrinya it succeeded on 93%.

In all cases, the errors were the result of missing roots in the lexicon or bugs in the implementation of specific phonological rules. These deficiencies have been fixed in the most recent version of the program.

Although more testing is called for, this evaluation suggests excellent coverage of Amharic and Tigrinya verbs for which the roots are known. Verbs are the source of most of the morphological complexity in these languages. Nouns and adjectives, the only other words calling for morphological analysis, are considerably simpler. Because the plural of Tigrinya nouns is usually not predictable, and we have access to only limited lexical resources for the language, we have not yet incorporated noun analysis and generation for that language. For Amharic, however, the system is apparently able to at least analyze the great majority of nouns and adjectives. We treat all Amharic words other than verbs, nouns, and adjectives as unanalyzed lexemes.

For Oromo, the newest language handled by HornMorpho, we have not yet conducted a comparable evaluation. Any evaluation of Oromo is complicated by the great variation in the use of double consonants and vowels by Oromo writers. We have two alternatives for evaluation: either we make the analyzer more lenient so that it accepts both single and double vowels and consonants in particular contexts or we restrict the evaluation to texts that have been verified to conform to particular orthographic standards.

5 Conclusions and ongoing work

For languages with complex morphology, such as Amharic, Tigrinya, and Oromo, almost all computational work depends on the existence of tools for morphological processing. HornMorpho is a

[3] The Amharic noun generator has not yet been evaluated.

first step in this direction. The goal is software that serves the needs of developers, and it is expected that the system will evolve as it is used for different purposes. Indeed, some features of the Amharic component of the system have been added in response to requests from users.

One weakness of the present system results from the limited number of available roots and stems, especially in the case of Tigrinya. When a root is not known, the Tigrinya verb guesser analyzer produces as many as 15 different analyses, when in many cases only one of these contains a root that actually exists in the language. However, the guesser analyzer itself is a useful tool for extending the lexicon; when an unfamiliar root is found in multiple wordforms and in multiple morphological environments, it can be safely added to the root lexicon. We have explored this idea elsewhere (Gasser, 2010).

A more significant weakness of the analyzers for all three languages is the handling of ambiguity. Even when a root or stem is known, there are often multiple analyses, and the program provides no information about which analyses are more likely than others. We are currently working on extending the weighted FST framework to accommodate probabilities as well as feature structures on transitions so that analyses can be ranked for their likelihood.

Although Amharic and Tigrinya have very similar verb morphology, they are handled by completely separate FSTs in the current implementation. In future work we will be addressing the question of how to share components of the system across related languages and how to build on existing resources to extend the system to handle related Semitic (e.g., Tigre, Silt'e) and Cushitic (e.g., Somali, Sidama) languages of the region.

Finally, HornMorpho is designed with developers in mind, people who are likely to be comfortable interacting with the program through the Python interpreter. However, morphological analysis and generation could also be of interest to the general public, including those who are learning the languages as second languages. We are currently experimenting with more user-friendly interfaces. As an initial step, we have created a web application for analyzing and generating Tigrinya verbs, which is available here: `http://www.cs.indiana.edu/cgi-pub/gasser/L3/morpho/Ti/v/anal/`.

References

Aklilu, A. (1987). *Amharic-English Dictionary*. Kuraz Printing Press, Addis Ababa.

Amsalu, S. and Demeke, G. A. (2006). Non-concatenative finite-state morphotactics of Amharic simple verbs. *ELRC Working Papers*, 2(3).

Amtrup, J. (2003). Morphology in machine translation systems: Efficient integration of finite state transducers and feature structure descriptions. *Machine Translation*, 18:213–235.

Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology*. CSLI Publications, Stanford, CA, USA.

Bitima, T. (2000). *A dictionary of Oromo technical terms*. Rüdiger Köpper Verlag, Köln.

Firdyiwek, Y. and Yaqob, D. (1997). The system for Ethiopic representation in ASCII. URL: citeseer.ist.psu.edu/56365.html.

Gasser, M. (2009). Semitic morphological analysis and generation using finite state transducers with feature structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 309–317, Athens, Greece.

Gasser, M. (2010). Expanding the lexicon for a resource-poor language using a morphological analyzer and a web crawler. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Gragg, G. (1982). *Oromo dictionary*. Michigan State University Press, East Lansing, MI, USA.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, 20:331–378.

Karttunen, L., Kaplan, R. M., and Zaenen, A. (1992). Two-level morphology with composition. In *Proceedings of the International Conference on Computational Linguistics*, volume 14, pages 141–148.

Koskenniemi, K. (1983). Two-level morphology: a general computational model for word-form recognition and production. Technical Report Publication No. 11, Department of General Linguistics, University of Helsinki.

Zacarias, E. (2009). Memhir.org dictionaries (English-Tigrinya, Hebrew-Tigrinya dictionaries). Available at http://www.memhr.org/dic/.