Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming

Wondwossen Mulugeta[1] and Michael Gasser[2]
[1]Addis Ababa University, Addis Ababa, Ethiopia
[2]Indiana University, Bloomington, USA
E-mail: [1]wondgewe@indiana.edu, [2]gasser@cs.indiana.edu

Abstract

This paper presents a supervised machine learning approach to morphological analysis of Amharic verbs. We use Inductive Logic Programming (ILP), implemented in CLOG. CLOG learns rules as a first order predicate decision list. Amharic, an under-resourced African language, has very complex inflectional and derivational verb morphology, with four and five possible prefixes and suffixes respectively. While the affixes are used to show various grammatical features, this paper addresses only subject prefixes and suffixes. The training data used to learn the morphological rules are manually prepared according to the structure of the background predicates used for the learning process. The training resulted in 108 stem extraction and 19 root template extraction rules from the examples provided. After combining the various rules generated, the program has been tested using a test set containing 1,784 Amharic verbs. An accuracy of 86.99% has been achieved, encouraging further application of the method for complex Amharic verbs and other parts of speech.

1. Introduction

Amharic is a Semitic language, related to Hebrew, Arabic, and Syriac. Next to Arabic, it is the second most spoken Semitic language with around 27 million speakers (Sieber, 2005; Gasser, 2011). As the working language of the Ethiopian Federal Government and some regional governments in Ethiopia, most documents in the country are produced in Amharic. There is also an enormous production of electronic and online accessible Amharic documents.

One of the fundamental computational tasks for a language is analysis of its morphology, where the goal is to derive the root and grammatical properties of a word based on its internal structure. Morphological analysis, especially for complex languages like Amharic, is vital for development and application of many practical natural language processing systems such as machine-readable dictionaries, machine translation, information retrieval, spell-checkers, and speech recognition.

While various approaches have been used for other languages, Amharic morphology has so far been attempted using only rule-based methods. In this paper, we applied machine learning to the task.

2. Amharic Verb Morphology

The different parts of speech and their formation along with the interrelationships which constitute the morphology of Amharic words have been more or less thoroughly studied by linguists (Sieber, 2005; Dwawkins, 1960; Bender, 1968). In addition to lexical information, the morphemes in an Amharic verb convey subject and object person, number, and gender; tense, aspect, and mood; various derivational categories such as passive, causative, and reciprocal; polarity (affirmative/negative); relativization; and a range of prepositions and conjunctions.

For Amharic, like most other languages, verbs have the most complex morphology. In addition to the affixation, reduplication, and compounding common to other languages, in Amharic, as in other Semitic languages, verb stems consist of a root + vowels + template merger (e.g., *sbr* + ee + CVCVC, which leads to the stem *seber* ሰበር[1] 'broke') (Yimam, 1995; Bender, 1968). This non-concatenative process makes morphological analysis more complex than in languages whose morphology is characterized by simple affixation. The affixes also contribute to the complexity. Verbs can take up to four prefixes and up to five suffixes, and the affixes have an intricate set of co-occurrence rules.

For Amharic verbs, grammatical features are not only shown using the affixes. The intercalation pattern of the consonants and the vowels that make up the verb stem will also be used to determine various grammatical features of the word. For example, the following two words have the same prefixes and suffixes and the same root while the pattern in which the consonants and the vowels intercalated is different, resulting in different grammatical information.

*?-**sebr**-alehu→ 1s pers.sing. simplex imperfective*
*?-**seber**-alehu→ 1st pers.sing.passive imperfective*

Figure 1: Stem template variation example

In this second case, the difference in grammatical feature is due to the affixes rather than the internal root template structure of the word.

*te-**deres**-ku → 1st pers. sing. passive perfective*
***deres**-ku → 1st pers. sing. simplex perfective*

Figure 2: Affix variation example

[1] *Amharic is written in the Geez writing system. For our morphology learning system we romanize Amharic orthography, and we cite these romanized forms in this paper.*

As in many other languages, Amharic morphology is also characterized by alternation rules governing the form that morphemes take in particular environments. The alternation can happen either at the stem affix intersection points or within the stem itself. Suffix-based alternation is seen, for example, in the second person singular feminine imperfect and imperative, shown in Table 1. The first two examples in Table 1 shows that, the second person singular feminine imperative marker *'i'*, if preceded by the character *'l'*, is altered to *'y'*. The last two examples show that the same alternation rule applies for imperfect roots.

No.	Word	Root	Feature
1	**gdel**	gdl	2nd person sing. masc. imperative
2	**gdey (gdel-i)**	gdl	2nd person sing. fem. imperative
3	**t-gedl-aleh**	gdl	2nd person sing. masc. imperfect
4	**t-gedy-alex**	gdl	2nd person sing. fem. imperfect

Table 1: Example of Amharic Alternation Rule

3. Machine Learning of Morphology

Since Koskenniemi's (1983) ground-breaking work on two-level morphology, there has been a great deal of progress in finite-state techniques for encoding morphological rules (Beesley & Karttunen, 2003). However, creating rules by hand is an arduous and time-consuming task, especially for a complex language like Amharic. Furthermore, a knowledge-based system is difficult to debug, modify, or adapt to other similar languages. Our experience with HornMorpho (Gasser, 2011), a rule-based morphological analyser and generator for Amharic, Oromo, and Tigrinya, confirms this. For these reasons, there is considerable interest in robust machine learning approaches to morphology, which extract linguistic knowledge automatically from an annotated or un-annotated corpus. Our work belongs to this category.

Morphology learning systems may be unsupervised (Goldsmith, 2001; Hammarström & Borin, 2011; De Pauw & Wagacha, 2007) or supervised (Oflazer *et al* 2001; Kazakov, 2000). Unsupervised systems are trained on unprocessed word forms and have the obvious advantage of not requiring segmented data. On the other hand, supervised approaches have important advantages of their own where they are less dependent on large corpora, requires less human effort, relatively fast which makes it scalable to other languages and that all rules in the language need not be enumerated.

Supervised morphology learning systems are usually based on two-level morphology. These approaches differ in the level of supervision they use to capture the rules. A weakly supervised approach uses word pairs as input (Manandhar *et al*, 1998; Mooney & Califf, 1995; Zdravkova *et al*, 2005). Other systems may require segmentation of input words or an analysis in the form of a stem or root and a set of grammatical morphemes.

4. ILP and Morphology Learning

Inductive Logic Programming (ILP) is a supervised machine learning framework based on logic programming. In ILP a hypothesis is drawn from background knowledge and examples. The examples (E), background knowledge (B) and hypothesis (H) all take the form of logic programs. The background knowledge and the final hypothesis induced from the examples are used to evaluate new instances.

Since logic programming allows for the expression of arbitrary relations between objects, ILP is more expressive than attribute-value representations, enabling flexible use of background knowledge (Bratko & King, 1994; Mooney & Califf, 1995). It also has advantages over approaches such as n-gram models, Hidden Markov Models, neural networks and SVM, which represent examples using fixed length feature vectors (Bratko & King, 1994). These techniques have difficulty representing relations, recursion and unbounded structural representation (Mooney, 2003). ILP, on the other hand, employs a rich knowledge representation language without length constraints. Moreover, the first order logic that is used in ILP limits the amount of feature extraction required in other approaches.

In induction, one begins with some data during the training phase, and then determines what general conclusion can logically be derived from those data. For morphological analysis, the learning data would be expected to guide the construction of word formation rules and interactions between the constituents of a word.

There have been only a few attempts to apply ILP to morphology, and most of these have dealt with languages with relatively simple morphology handling few affixations (Kazakov, 2000; Manandhar et al, 1998; Zdravkova et al, 2005). However, the results are encouraging.

While we focus on Amharic verb morphology, our goal is a general-purpose ILP morphology learner. Thus we seek background knowledge that is plausible across languages that can be combined with language-specific examples to yield rule hypotheses that generalize to new examples in the language.

CLOG is a Prolog based ILP system, developed by Manandhar *et al* (1998)[2], for learning first order decision lists (rules) on the basis of positive examples only. A rule in Prolog is a clause with one or more conditions. The right-hand side of the rule (the body) is a condition and the left-hand side of the rule (the head) is the conclusion. The operator between the left and the right hand side (the sign '**:-**') means ***if***. The body of a rule is a list of goals separated by commas, where commas are understood as conjunctions. For a rule to be true, all of its conditions/goals must be evaluated to be true. In the expression below, ***p*** is true if ***q*** and ***r*** are true or if ***s*** and ***t*** are true.

[2] *CLOG is freely available ILP system at: http://www-users.cs.york.ac.uk/suresh/CLOG.html)*

$$\left.\begin{array}{l} p \text{ :- } q, r. \\ p \text{ :- } s, t. \end{array}\right\} \; p \Leftrightarrow (q \wedge r) \vee (s \wedge t)$$

*Where **q**, **r**, **s** and **t** could be facts or predicates with any arity and **p** is a predicate with any number of arguments.*

CLOG relies on output completeness, which assumes that every form of an object is included in the example and everything else is excluded (Mooney & Califf, 1995). We preferred CLOG over other ILP systems because it requires only positive examples and runs faster than the other variants (Manandhar *et al*, 1998). CLOG uses a hill climbing strategy to build the rules, starting from a simple goal and iteratively adding more rules to satisfy the goal until there are no possible improvements. The evaluation of the rules generated by the learner is validated using a gain function that compares the number of positively and negatively covered examples in the current and previous learning stages (Manandhar et al, 1998).

5. Experiment Setup and Data

Learning morphological rules with ILP requires preparation of the training data and background knowledge. To handle a language of the complexity of Amharic, we require background knowledge predicates that can handle stem extraction by identifying affixes, root and vowel identification and grammatical feature association with constituents of the word.

The training data used during the experiment is of the following form:

```
stem([s,e,b,e,r,k,u],[s,e,b,e,r],[s,b,r] [1,1]).
stem([s,e,b,e,r,k],[s,e,b,e,r],[s,b,r], [1,2]).
stem([s,e,b,e,r,x],[s,e,b,e,r],[s,b,r], [1,3]).
```

Figure 3: Sample examples for stem and root learning

The predicate *'stem'* provides a word and its stem to permit the extraction of the affixes and root template structure of the word. The first three parameters specify the input word, the stem of the word after affixes are removed, and the root of the stem respectively. The fourth parameter is the codification of the grammatical features (tense-aspect-mood and subject) of the word.

Taking the second example in Figure 3, the word ***seberk*** has the stem ***seber*** with the root ***sbr*** and is perfective (the first element of the third parameter which is 1) with second person singular masculine subject (the second element of the third parameter is 2).

We codified the grammatical features of the words and made them parameters of the training data set rather than representing the morphosyntactic description as predicates as in approaches used for other languages (Zdravkova et al, 2005).

The background knowledge also includes predicates for string manipulation and root extraction. Both are language-independent, making the approach adaptable to other similar languages. We run three separate training experiments to learn the stem extraction, root patterns, and internal stem alternation rules.

a) Learning stem extraction:
The background predicate *'set_affix'* uses a combination of multiple '*split'* operations to identify the prefix and suffixes attached to the input word. This predicate is used to learn the affixes from examples presented as in Figure 3 by taking only the *Word* and the *Stem* (the first two arguments from the example).

```
set_affix(Word, Stem, P1,P2,S1,S2):-
   split(Word, P1, W11),
   split(Stem, P2, W22),
   split(W11, X, S1),
   split(W22, X, S2),
   not( (P1=[],P2=[],S1=[],S2=[])).
```

Figure 4: Affix extraction predicate

The predicate makes all possible splits of *Word* and *Stem* into three segments to identify the prefix and suffix substitutions required to unify *Stem* with *Word*. In this predicate, P1 and S1 are the prefix and suffix of the *Word*; while P2 and S2 are the prefix and suffix of the *Stem* respectively. For example, if *Word* and *Stem* are ***tgedyalex*** and ***gedl*** respectively, then the predicate will try all possible splits, and one of these splits will result in P1=[***t***], P2=[], S1=[***yalex***] and S2=[***l***]. That is, ***tgedyalex*** will be associated with the stem ***gedl***, if the prefix P1 is replaced with P2 and the suffix S1is replaced with S2.
The ultimate objective of this predicate is to identify the prefix and suffix of a word and then extract the valid stem (*Stem*) from the input string (*Word*).
Here, we have used the utility predicate '*split'* that segments any input string into all possible pairs of substrings. For example, the string ***sebr*** could be segmented as {([]-[*sebr*]), ([*s*]-[*ebr*]), ([*se*]-[*br*]), ([*seb*]-[*r*]), or ([*sebr*]-[])}.

b) Learning Roots:
The root extraction predicate, '*root_vocal'*, extracts *Root* and the *Vowel* with the right sequence from the *Stem*. This predicate learns the root from examples presented as in Figure 3 by taking only the *Stem* and the *Root* (the second and third arguments).

```
root_vocal(Stem,Root,Vowel):-
       merge(Stem,Root,Vowel).

merge([X,Y,Z|T],[X,Y|R],[Z|V]):-
              merge(T,R,V).
merge([X,Y|T],R,[X,Y|V]):-
              merge(T,R,V).
merge([X|Y],[X|Z],W) :-
              merge(Y,Z,W).
merge([X|Y],Z,[X|W]) :-
              merge(Y,Z,W).
```

Figure 5: Root template extraction predicate

The predicate '*root_vocal'* performs unconstrained permutation of the characters in the *Stem* until the first part of the permutated string matches the *Root* character pattern provided during the training. The goal of this predicate is to separate the vowels and the consonants of a *Stem*. In this predicate we have used the utility predicate '*merge'* to perform the permutation. For example, if *Stem* is ***seber*** and the example associates this stem with the *Root* **sbr**, then *'root_temp'*, using '*merge',* will generate many patterns, one of which would be ***sbree***. This, ultimately, will learn that the vowel pattern [***ee***] is valid within a stem.

c) Learning stem internal alternations:
Another challenge for Amharic verb morphology learning is handling stem internal alternations. For this purpose, we have used the background predicate *'set_internal_alter'*:

```
set_internal_alter(Stem,Valid_Stem,St1,St2):-
            split(Stem,P1,X1),
            split(Valid_Stem,P1,X2),
            split(X1,St1,Y1),
            split(X2,St2,Y1).
```

Figure 6: stem internal alternation extractor

This predicate works much like the '*set_affix*' predicate except that it replaces a substring which is found in the middle of *Stem* by another substring from *Valid_Stem*. In order to learn stem alternations, we require a different set of training data showing examples of stem internal alternations. Figure 7 shows some sample examples used for learning such rules.

```
alter([h,e,d],[h,y,e,d]).
alter([m,o,t],[m,e,w,o,t]).
alter([s,a,m],[s,e,?,a,m]).
```

Figure 7: Examples for internal stem alternation learning

The first example in Figure 7 shows that for the words ***hed*** and ***hyed*** to unify, the ***e*** in the first argument should be replaced with ***ye***.

Along with the three experiments for learning various aspects of verb morphology, we have also used two utility predicates to support the integration between the learned rules and to include some language specific features. These predicates are '*template'* and '*feature'*:

➢ '*template'*: used to extract the valid template for *Stem*. The predicate manipulates the stem to identify positions for the vowels. This predicate uses the list of vowels (vocal) in the language to assign '0' for the vowels and '1' for the consonants.

```
template([],[]).
template([X|T1],[Y|B]):-
        template(T1,B),
        (vocal(X)->Y=0;Y=1).
```

Figure 8: CV pattern decoding predicate

For the stem ***seber*** this predicate tries each character separately and finally generates the pattern [1,0,1,0,1] and for the stem ***sebr***, it generates [1,0,1,1] to show the valid template of Amharic verbs.

➢ '*feature'*: used to associate the identified affixes and root CV pattern with the known grammatical features from the example. This predicate uses a codified representation of the eight subjects and four tense-aspect-mood features ('tam') of Amharic verbs, which is also encoded as background knowledge. This predicate is the only language-dependent background knowledge we have used in our implementation.

```
feature([X,Y],[X1,Y1]):-
        tam([X],X1),
        subj([Y],Y1).
```

Figure 9: Grammatical feature assignment predicate

6. Experiments and Result

For CLOG to learn a set of rules, the predicate and arity for the rules must be provided. Since we are learning words by associating them with their stem, root and grammatical features, we use the predicate schemas ***rule(stem(_,_,_,_))*** for *set_affix* and *root_vocal*, and ***rule(alter(_,_))*** for *set_internal_alter*. The training examples are also structured according to these predicate schemas.

The training set contains 216 manually prepared Amharic verbs. The example contains all possible combinations of tense and subject features. Each word is first romanized, then segmented into the stem and grammatical features, as required by the '*stem'* predicate in the background knowledge. When the word results from the application of one or more alternation rules, the stem appears in the canonical form. For example, for the word ***gdey***, the stem specified is ***gdel*** (see the second example in Table 1).

Characters in the Amharic orthography represent syllables, hiding the detailed interaction between the consonants and the vowels. For example, the masculine imperative verb 'ግደል' ***gdel*** can be made feminine by adding the suffix 'i' (gdel-i). But, in Amharic, when the dental 'l' is followed by the vowel 'i', it is palatalized, becoming 'y'. Thus, the feminine form would be written 'ግደይ', where the character 'ይ' 'y' corresponds to the sequence 'l-i'.

To perform the romanization, we have used our own Prolog script which maps Amharic characters directly to sequences of roman consonants and vowels, using the familiar SERA transliteration scheme. Since the mapping is reversible, it is straightforward to convert extracted forms back to Amharic script.

After training the program using the example set, which took around 58 seconds, 108 rules for affix extraction, 18 rules for root template extraction and 3 rules for internal stem alternation have been learned. A sample rule generated for affix identification and associating the word constituents with the grammatical features is shown below:

```
stem(Word, Stem, [2, 7]):-
        set_affix(Word, Stem, [y], [], [u], []),
        feature([2, 7], [imperfective, tppn]),
        template(Stem, [1, 0, 1, 1]).
```

Figure 10: Learned affix identification rule example

The above rule declares that, if the word starts with ***y*** and ends with ***u*** and if the stem extracted from the word after stripping off the affixes has a CVCC ([1,0,1,1]) pattern, then that word is imperfective with third person plural neutral subject (tppn).

```
alter(Stem,Valid_Stem):-
  set_internal_alter(Stem,Valid_Stem, [o], [e, w, o]).
```

Figure 11: Learned internal alternation rule example

The above rule will make a substitution of the vowel ***o*** in a specific circumstances (which is included in the program) with ***ewo*** to transform the initial stem to a valid stem in the language. For example, if the Stem is ***zor***, then ***o*** will be replaced with ***ewo*** to give ***zewor.***

The other part of the program handles formation of the root of the verb by extracting the template and the vowel sequence from the stem. A sample rule generated to handle the task looks like the following:

```
root(Stem, Root):-
        root_vocal(Stem, Root, [e, e]),
         template(Stem, [1, 0, 1, 0, 1]) .
```

Figure 12: Learned root-template extraction rule example

The above rule declares that, as long as the consonant vowel sequence of a word is CVCVC and both vowels are *e*, the stem is a possible valid verb. Our current implementation does not use a dictionary to validate whether the verb is an existing word in Amharic.

Finally, we have combined the background predicates used for the three learning tasks and the utility predicates. We have also integrated all the rules learned in each experiment with the background predicates. The integration involves the combination of the predicates in the appropriate order: stem analysis followed by internal stem alternation and root extraction.

After building the program, to test the performance of the system, we started with verbs in their third person singular masculine form, selected from the list of verbs transcribed from the appendix of Armbruster (1908)[3]. We then inflected the verbs for the eight subjects and four tense-aspect-mood features of Amharic, resulting in 1,784 distinct verb forms. The following are sample analyses of new verbs that are not part of the training set by the program:

```
InputWord: [a, t, e, m, k, u]
      Stem: [?, a, t, e, m]
      Template: [1,0, 1, 0, 1]
      Root: [?, t, m]
      GrammaticalFeature: [perfective, fpsn*]
```

Figure 13: Sample Test Result (with boundary alternation)

*fpsn: first person singular neuter

[3] *Available online at:* http://nlp.amharic.org/resources/lexical/word-lists/verbs/c-h-armbruster-initia-amharica/ *(accessed February 12, 2012).*

The above example shows that the suffix that needs to be stripped off is ***[k,u]*** and that there is an alternation rule that changes ***'a'*** to ***'?,a'*** at the beginning of the word.

```
InputWord: [t, k, e, f, y, a, l, e, x]
     Stem: [k, e, f, l]
     Template: [1,0, 1, 1]
     Root: [k, f, l]
     GrammaticalFeature: [imperfective, spsf*]
```

Figure 14: Sample Test Result (Internal alternation)

*spsf: second person singular feminine

The above example shows that the prefix and suffix that need to be stripped off are ***[t]*** and ***[a,l,e,x]*** respectively and that there is an alternation rule that changes ***'y'*** to ***'l'*** at the end of the stem after removing the suffix.

The system is able to correctly analyze 1,552 words, resulting in 86.99% accuracy. With the small set of training data, the result is encouraging and we believe that the performance will be enhanced with more training examples of various grammatical combinations.

The wrong analyses and test cases that are not handled by the program are attributed to the absence of such examples in the training set and an inappropriate alternation rule resulting in multiple analysis of a single test word.

Test Word	***Stem***	***Root***	***Feature***
[s,e,m,a,c,h,u]	*[s,e,m,a,?]*	*[s,m,?]*	*perfective, sppn*
[s,e,m,a,c,h,u]	*[s,e,y,e,m]*	*[s,y,m]*	*gerundive, sppn*
[l,e,g,u,m,u]	*[l,e,g,u,m]*	*NA*	*NA*

Table 2: Example of wrong analysis

Table 2 shows some of the wrong analyses and words that are not analyzed at all. The second example shows that an alternation rules has been applied to the stem resulting in wrong analysis (the stem should have been the one in the first example). The last example generated a stem with vowel sequence of '***eu'*** which is not found in any of the training set, categorizing the word in the not-analyzed category.

7. Future work

ILP has proven to be applicable for word formation rule extraction for languages with simple rules like English. Our experiment shows that the approach can also be used for complex languages with more sophisticated background predicates and more examples. While Amharic has more prefixes and suffixes for various morphological features, our system is limited to only subject markers. Moreover, all possible combinations of subject and tense-aspect-mood have been provided in the training examples for the training. This approach is not practical if all the prefix and suffixes are going to be included in the learning process.

One of the limitations observed in ILP for morphology learning is the inability to learn rules from incomplete examples. In languages such as Amharic, there is a range of complex interactions among the different morphemes, but we cannot expect every one of the thousands of morpheme combinations to appear in the training set. When examples are limited to only some of the legal morpheme combinations, CLOG is inadequate because it is not able to use variables as part of the body of the predicates to be learned.

An example of a rule that could be learned from partial examples is the following: *“if a word has the prefix* ***'te',*** *then the word is passive no matter what the other morphemes are”*. This rule (not learned by our system) is shown in Figure 15.

```
stem(Word, Stem, Root, GrmFeatu):-
        set_affix(Word, Stem, [t,e], [], S, []),
        root_vocal(Stem, Root, [e, e]),
        template(Stem, [1, 0, 1, 0, 1]),
        feature(GrmFeatu, [Ten, passive, Sub]).
```

Figure 15: Possible stem analysis rule with partial feature

That is, ***S*** is one of the valid suffixes, ***Ten*** is the Tense, and ***Sub*** is the subject, which can take any of the possible values.

Moreover, as shown in section 2, in Amharic verbs, some grammatical information is shown by various combinations of affixes. The various constraints on the co-occurrence of affixes are the other problem that needs to be tackled. For example, the 2nd person masculine singular imperfective suffix ***aleh*** can only co-occur with the 2nd person prefix ***t*** in words like ***t-sebr-aleh***. At the same time, the same prefix can occur with the suffix ***alachu*** for the 2nd person plural imperfective form. To represent these constraints, we apparently need explicit predicates that are specific to the particular affix relationship. However, CLOG is limited to learning only the predicates that it has been provided with.

We are currently experimenting with genetic programming as a way to learn new predicates based on the predicates that are learned using CLOG.

8. Conclusion

We have shown in this paper that ILP can be used to fast-track the process of learning morphological rules of complex languages like Amharic with a relatively small number of examples. Our implementation goes beyond simple affix identification and confronts one of the challenges in template morphology by learning the root-template extraction as well as stem-internal alternation rule identification exhibited in Amharic and other Semitic languages. Our implementation also succeeds in learning to relate grammatical features with word constituents.

9. References

Armbruster, C. H. (1908). *Initia Amharic: an Introduction to Spoken Amharic.* Cambridge: Cambridge University Press.

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

Bender, M. L. (1968). *Amharic Verb Morphology: A Generative Approach.* Ph.D. thesis, Graduate School of Texas.

Bratko, I. and King, R. (1994). *Applications of Inductive Logic Programming*. *SIGART Bull. 5*, 1, 43-49.

Dawkins, C. H., (1960). *The Fundamentals of Amharic*. Sudan Interior Mission, Addis Ababa, Ethiopia.

De Pauw, G. and P.W. Wagacha. *(2007). Bootstrapping Morphological Analysis of Gĩkũyũ Using Unsupervised Maximum Entropy Learning.* Proceedings of the Eighth INTERSPEECH Conference, Antwerp, Belgium.

Gasser, M. (2011). *HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya*. Conference on Human Language Technology for Development, Alexandria, Egypt.

Goldsmith, J. (2001). *The unsupervised learning of natural language morphology*. Computational Linguistics, 27: 153-198.

Hammarström, H. and L. Borin. (2011). *Unsupervised learning of morphology*. Computational Linguistics, 37(2): 309-350.

Kazakov, D. (2000). *Achievements and Prospects of Learning Word Morphology with ILP*, Learning Language in Logic, Lecture Notes in Computer Science.

Kazakov, D. and S. Manandhar. (2001). *Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming*. Machine Learning, 43:121–162.

Koskenniemi, K. (1983). *Two-level Morphology: a General Computational Model for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki, Technical Report No. 11.

Manandhar, S. , Džeroski, S. and Erjavec, T. (1998). *Learning multilingual morphology with CLOG.* Proceedings of Inductive Logic Programming. 8th International Workshop in Lecture Notes in Artificial Intelligence. Page, David (Eds) pp.135–44. Berlin: Springer-Verlag.

Mooney, R. J. (2003). *Machine Learning*. Oxford Handbook of Computational Linguistics, Oxford University Press, pp. 376-394.

Mooney, R. J. and Califf, M.E. (1995). *Induction of first-order decision lists: results on learning the past tense of English verbs*, Journal of Artificial Intelligence Research, v.3 n.1, p.1-24.

Oflazer, K., M. McShane, and S. Nirenburg. (2001). *Bootstrapping morphological analyzers by combining human elicitation and machine learning*. Computational Linguistics, 27(1):59–85.

Sieber, G. (2005). *Automatic Learning Approaches to Morphology*, University of Tübingen, International Studies in Computational Linguistics.

Yimam, B. (1995). *Yamarigna Sewasiw (Amharic Grammar).* Addis Ababa: EMPDA.

Zdravkova, K., A. Ivanovska, S. Dzeroski and T. Erjavec, (2005). *Learning Rules for Morphological Analysis and Synthesis of Macedonian Nouns*. In Proceedings of SIKDD 2005, Ljubljana.