Incremental Learning of Affix Segmentation

Wondwossen Mulugeta[1], *Michael Gasser*[2], *Baye Yimam*[1]
(1) ADDIS ABABA UNIVERSITY, Addis Ababa, Ethiopia
(2) INDIANA UNIVERSITY, Bloomington, USA
wondisho@aau.edu.et, gasser@cs.indiana.edu, bayemekonnen@gmail.com

ABSTRACT

This paper presents a supervised machine learning approach to incrementally learn and segment affixes using generic background knowledge. We used Prolog script to split affixes from the Amharic word for further morphological analysis. Amharic, a Semitic language, has very complex inflectional and derivational verb morphology, with many possible prefixes and suffixes which are used to show various grammatical features. Further segmentation of the affixes into valid morphemes is a challenge addressed in this paper. The paper demonstrates how incremental and easy-to-complex examples can be used to learn such language constructs. The experiment revealed that affixes could be further segmented into valid prefixes and suffixes using a generic and robust string manipulation script by the help of an intelligent teacher who presents examples in incremental order of complexity allowing the system to gradually build its knowledge. The system is able to do the segmentation with 0.94 Precision and 0.97 Recall rates.

KEYWORDS: Amharic, Morphology, Segmentation, Incremental Learning, ILP, Machine Learning

Proceedings of COLING 2012: Technical Papers, pages 1901–1914,
COLING 2012, Mumbai, December 2012.

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[1], 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 our previous work, we have tried to apply a machine learning approach to learn morphological rules. In the experiment, we were able to learn various affixes attached to the stem and analyze the internal stem structure of the verb which is one crucial task in Semitic morphology. The major limitation of the work concerns words made up of the stem and more than one adjacent prefix or suffix; in those cases the system fails to segment the affixes. This work presents the continuation of our previous system and attempts to further segment the affixes into valid prefixes and suffixes using generic and incremental learning without any initial knowledge of the prefixes and suffixes of the language.

2 Amharic Verb Morphology and Affixation

Amharic, with all its complex word formation nature, has been more or less thoroughly studied by linguists (Sieber, 2005; Dawkins, 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.

2.1 Amharic Verb Morphology

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[2] 'broke') (Yimam, 1995; Bender, 1968). This non-concatenative process makes morphological analysis more

[1] Some of these are: Addis Ababa City Council, Amhara Region, Benishangul-Gumuz Region and Dire Dawa Administrative Council

[2] 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.

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.

Grammatical features on Amharic verbs 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. For example, the following two verbs have the same prefixes and suffixes and the same root while the pattern in which the consonants and the vowels intercalate is different, resulting in different grammatical information.

*?-**sebr**-alehu* (እሰብራለሁ)→ *1s pers. sing. simplex imperfective*
Gloss*: 'I will break'*
*?-**seber**-alehu* (እሰበራለሁ)→ *1stpers.sing.passive imperfective*
Gloss*: 'I will be broken'*

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-**seber**-ku* (ተሰበርኩ)→ *1st pers. sing. passive perfective*
Gloss*: 'I was/have been broken'*
***seber**-ku* (ሰበርኩ)→ *1st pers. sing. simplex perfective*
Gloss*: 'I broke'*

FIGURE 2 – Affix variation example

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. Amharic is also characterized by alternation between morphemes of the affixes. For example, the prefix ***'ye'*** if it comes before the negative prefix ***'al'***, alternation occurs and the form changes to ***'yal'*** where the ***'e'*** sound gets deleted.

2.2 Affixation in Amharic

Languages having multiple morphemes concatenated to form prefixes and suffixes show some interrelationship and co-occurrence sequences. Affixes have predefined slots in a language. The slots constrain the occurrence of the affixes, and a generic morphological learning system should be flexible enough to learn the slots and the morphemes that can fill each of them. Such 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 unanalyzed word forms and have the obvious advantage of not requiring segmented data. The segmentation result will help to learn rules by using thin supervised examples.

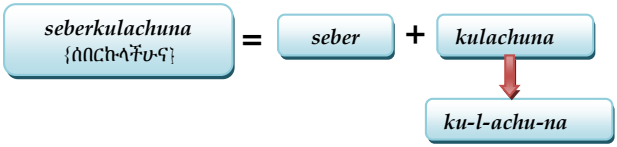

FIGURE 3 – Stem Suffix Analysis Example

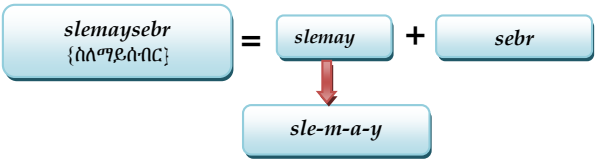

FIGURE 4 – Stem Prefix Analysis Example[3]

3 Incremental Affix Segmentation and ILP

Incremental learning dictates the use of less complex structures to be learned at early stages and move on to more complex and sophisticated structures using knowledge of previous structures as a basis. Such learning process can be implemented using Inductive Logic Programming (ILP) which is a machine learning approach that learns rules from positive and negative examples.

3.1 Incremental Learning

The problem of language acquisition has been one of the central issues in cognitive science, as well as in linguistics. Within the framework of Universal Grammar (Chomsky, 1981), language acquisition is assumed to be the process of setting the values of parameters, which are conceived of as innately-specified points of grammatical variations that have multiple consequences for the different aspects of the surface grammar. Here, Chomsky argued that language is so complex that the only way it could be learned is through innate constraints on what was a possible grammar using its parameters. More recently and in opposition to Chomsky's and others' innatist (or nativist) view, there have been arguments from psychologists and cognitive linguists who support empiricist theories of language acquisition. Among other things, they argue that innate constraints are not needed (this is one instance of the large "nature vs. nurture debate"). One of these simplifications come from what is "child-directed speech" (CDS), the simplified speech that adults naturally use when speaking to children. But of course adults adjust CDS as children get older, making it more and more complex. Thus, one argument claims that the only way children are able to learn language is through the graded simplification made by adults to the input that the child receives. This supports the idea of incremental learning, by which examples with less

[3] *The last two suffixes {a-y} are the result of the actual suffixes {al - y} transformed due to the assimilation process*

complex word structure are presented first and the knowledge of affixes acquired from early training becomes the basis of the more complex knowledge acquired later.

The ability to acquire and use language and its constructs is a key aspect that distinguishes humans from other beings. Learning is the process of acquiring knowledge over time from different realities we are exposed to. The same is true for acquiring language related facts and rules by human beings (Pirrelli & Herreros, 2005). The brain learns by observing, constantly labelling and creating its own rules that define or explain what has been observed. This learning process demands massive amount of data or exposure to relevant and interesting instances to deduce rules from. In cases where no such data is available or the aim is to learn from few examples, incremental learning through strategic example coverage would be suitable. Inspired by features of child language acquisition, the best way to learn language is by applying child language learning methods. Children observe and are able to identify similarities and add to their database such common features relating it to the meaning or the form of a word. For example, as presented by Sara Finley, when a child encounters words like {*dogs*, *cats*, *chairs*, *boys}*, he can discover part of the word forming feature similarity through the suffix 's', plural marker in this case (Finley, 2012). Thus, distributional cues are very important for children to find where relationship between words lie and find patterns for future use.

At early ages it is common to see children make such mistakes of segmenting part of the main word as affix or attaching affix on fully formed words. These actions are considered to be part of the learning process.

In morphology, learning constituents of a word in distributional or structural cues proved to be effective (Cavar, 2005) and be linked with incremental learning to teach the learner in a more logical manner. Such language processing by means of data-oriented methods emphasize the assumption that human language perception and production works with representations of concrete past language experiences, rather than with abstract grammar rules (Rens & Remko, 1996). The next section describes such incremental learning for affix segmentation task.

3.2 Incremental Affix Segmentation

Incremental learning of morphological affix segmentation results in knowledge acquisition when the system encounters new affixes, through the further segmentation of the string based on previous knowledge (Altenbek 2009). The first step in the segmentation process is to detach the affix from the main stem. This has been done using our previous system that employs inductive logic programming to learn stems and affixes as well as internal stem structure from examples (Mulugeta & Gasser, 2012). The system takes the main word and keeps database of valid affixes where the challenge relies on how to further analyze the affix into a valid list. In this regard, Prolog programming language is more suited for such action due to its easy knowledge acquisition and database manipulation features.

While we focus on Amharic verb affix segmentation and morphology learning, our goal is a general-purpose ILP morphology learner that automatically segments affixes based on previous knowledge during the learning process. Thus we seek background knowledge that is plausible across languages that can be combined with language-specific examples and

intelligent ways of presenting examples to yield rule hypotheses that generalize to new examples in the language.

3.3 Inductive Logic Programming

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

There have been only a few attempts to apply Inductive Logic Programming (ILP) to morphology. Most of these have dealt with languages with relatively simple morphology handling few affixations (Kazakov, 2000; Manandhar et al, 1998; Zdravkova et al, 2005). These attempts consider the affixes extracted as one singleton morpheme which is not the case for complex languages like Amharic. However, their results are found to be encouraging. The enhancement to such learning systems would be the task of further analysis of the affixes. The analysis shall include but not limited to prefix and suffix knowledge capturing through segmentation process to help build database of prefixes and suffixes for deep grammar scrutiny. This learning and knowledge acquisition task has been done using CLOG.

CLOG is a Prolog based ILP system, developed by Manandhar *et al* (1998)[4], 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 (the head) to be true, all of its conditions/goals must be evaluated to be true. In the expression below, there are two ways of evaluating the goal ***p*** even with two different results. Accordingly, ***p*** is true if ***q*** and ***r*** are true or if ***s*** and ***t*** are true[5].

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

*Where **q**, **r**, **s** and **t** could be facts or predicates 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).

---

[4] CLOG is a freely avalable ILP system at: (http://www-users.cs.york.ac.uk/suresh/CLOG.html)
[5] Refer to http://en.wikipedia.org/wiki/Prolog for detailed illustration on Prolog

4 Experiments and Integration with Morphology Learning System

ILP is a rarely used method for morphology and language related learning. The approach demands well crafted background knowledge with the level of depth required for supervision and a set of positive and/or negative examples to learn from. Our previous experiment, which also uses ILP, is able to extract verb morphological rules and internal stem structure as well as orthographic alternation rules from examples on Amharic subject markers (Mulugeta & Gasser, 2012). Our system takes examples of the form shown in Figure 5 and extract morphological rules based on the various generic background knowledge crafted for the learning task.

In Figure 5, 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 two parameters specify the input word and the stem of the word after affixes are removed. The third parameter is the codification of the grammatical features (tense-aspect-mood, voice, subject and object) of the word. The codification is a simple knowledge about the various grammatical features of the word. For example, the fourth value in the third argument represents the object marker of the word where 2 means second person singular masculine, 6 means third person plural neuter and so forth.

```
stem([s,e,b,e,r,k,u,l,h],[s,e,b,e,r] [1,1,1,2]).
stem([s,e,b,e,r,k,l,a,c,h,w],[s,e,b,e,r], [1,1,2,6]).
stem([s,e,b,e,r,x,l,n],[s,e,b,e,r], [1,1,3,8]).
```

FIGURE 5 – Sample training examples for the learning process

The background knowledge added to handle the affix segmentation and database manipulation is generic in its nature making it applicable for any language of interest. In addition, the background predicate also includes scripts for string manipulation and root extraction. Both are language-independent, making the approach adaptable to other similar languages.

The previous system is able to generate rules of the following structure by taking the examples of the form shown in Figure 5 above.

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

stem(Word, Stem, [2, 1, 1, 2]):-
        set_affix(Word, Stem, [te], [], [kulh], []),
        feature([2, 1,1, 2], [passive, perfective, fpsn, spsm]),
        template(Stem, [1, 0, 1, 0, 1]).
```

FIGURE 6 – Learned affix identification rule example

Accordingly, the rules learned through ILP contain three major background predicates:

- The ***'set_affix'*** predicate uses a combination of multiple 'split' operations to identify the prefixes and suffixes attached to the input word. This predicate is used to learn the affixes from examples presented by taking only the Word and the Stem parameters (the first two arguments from the example). The last four arguments of ***set_affix*** predicate represent the prefix and suffix pairs in the Word and Stem parameters.
- The ***'template'*** predicate is 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.
- The ***'feature'*** predicate is 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 grammatical features in the language, which is also encoded as background knowledge. This predicate is the only language-dependent background knowledge we have used in our implementation.

The two example rules in Figure 6 show that the prefixes *[y]* and *[te]* as well as the suffixes *[u]* and *[kulh]* are extracted from the examples with the respective root template structure. The output is limited with no further segmentation of the affixes to relate it with the grammatical features for further analysis. The current experiment includes a module which tries to do affix segmentation in incremental manner. The following algorithm and script presents how the segmentation is done in an incremental manner based on the experiment setup. While the algorithm is generic for any affix presented, the illustration shown later demonstrates that the order in which the examples are presented will dictate the knowledge acquired by the learner.

For each Suffix A extracted
    Take A as a possible Suffix
    Take any nonempty leftmost segment B of A
    Check if B exists in the Suffix database
    If B is a valid Suffix
        Remove A from the Suffix database
        Assign A to be the remaining substring
        Repeat the suffix segmentation process
End if there are no strings to segment

FIGURE 7 – Suffix segmentation algorithm

```
segS([ ]):-!.
segS(A):-
        findall(D, (append(C,D,A),C\==[ ],suffix(C),segS(D)), Segs),
        Segs==[ ]->assertz(suffix(A));!.

seg_suffix([ ],[ ]).
seg_suffix(A,[C|B]):-
        append(C,D,A),
        C\==[ ],
        suffix(C),
        seg_suffix(D,B).
```

FIGURE 8 – Suffix segmentation and list generation Prolog script

* ***segS*** *segments suffixes and updates the database whenever a new suffix is identified through the assertz predicate.*
* ***seg_suffix*** *convert one affix string into a list of morphemes enclosed in a square bracket. For example, [kulhna] will be changed to [[ku],[lh],[na]] based on prior affix knowledge.*

The system works progressively by picking examples from the training data and learning the affixes, stem template structure and alternation rules. Along with affix extraction, the systems takes each affix from the example and iteratively build its database and segment upcoming affixes based on this knowledge. The following analysis and knowledge acquisition example illustrates how the affix segmentation incrementally learns a suffix based on the examples presented.

Iteration 1
Initial Suffix Database: *{Ø}*
Training Example 1: *stem([s,e,b,e,r,k,u],[s,e,b,e,r] [1,1,1,0]).*
Stem Extraction Result: *[ ], [s,e,b,e,r], [k,u]*
Affix Identification*: [k,u] *no further segmentation as the database is empty*
Updated Suffix Database: *{ suffix([k,u])}*

Iteration 2
Initial Suffix Database: *{ suffix([k,u])}*
Training Example 2: *stem([s,e,b,e,r,k,u,l,h],[s,e,b,e,r] [1,1,1,2]).*
Stem Extraction Result: *[ ], [s,e,b,e,r], [k,u,l,h]*
Affix Identification*:{[k,u], [l,h]} *as [k,u] is already identified earlier*
Updated Suffix Database: *{ suffix([k,u]), suffix([l,h])}*

Iteration 3
Initial Suffix Database: *{ suffix([k,u]), suffix([l,h])}*
Training Example 2: *stem([s,e,b,e,r,k,u,l,h,n,a],[s,e,b,e,r] [1,1,1,2]).*
Stem Extraction Result: *[ ], [s,e,b,e,r], [k,u,l,h,n,a]*
Affix Identification*:{[k,u],[l,h],[n,a]} *as [k,u] and [l,h]are already identified earlier*
Updated Suffix Database: *{ suffix([k,u]), suffix([l,h]), suffix([n,a])}*

Iteration 4
Initial Suffix Database: *{ suffix([k,u]), suffix([l,h]), suffix([n,a])}}*
Training Example 2: *stem([s,e,b,e,r,k,u,l,a,c,h,u],[s,e,b,e,r] [1,1,1,6]).*
Stem Extraction Result: *[ ], [s,e,b,e,r], [k,u,l,h,n,a]*
Affix Identification*:{[k,u],[l,a,c,h,u]} *as [k,u] is already identified earlier*
Updated Suffix Database: *{ suffix([k,u]), suffix([l,h]), suffix([n,a]), suffix([l,a,c,h,u])}*

FIGURE 9 – Suffix learning and segmentation process illustration

The illustration in Figure 9 (only suffix learning as a show case) shows that, at the beginning of the learning process (iteration 1), the prefix and suffix database is empty assuming that the first examples to be presented shall have only a single prefix and suffix. These prefixes and suffixes shall be taken as the primary knowledge acquired. The upcoming and remaining affixes shall be learned based on this previous knowledge. After iteration 4, four suffixes are leaned that are results of incremental learning starting from an empty database and systematic presentation of examples. It should be noted here that if the example in iteration 2 has been presented first, the initial database would have been *{suffix([k,u,l,h])}* and the first example with the suffix *[k,u]* would not have any impact on its previous knowledge of suffix. This indicates that the system requires the skill of the teacher to guide the learner with more logical flow.

One potential drawback of such incremental learning is the need for an intelligent teacher. What does our teacher need to know? The most important requirement is information about the number of prefixes and suffixes that a word contains. The main constraint in presentation of words is the order of the example based on the number of affixes it contains. We would expect a literate native speaker of the language with a little linguistic training to have this awareness. One advantage of the current implementation is that, the rules learned with the segmented affixes are easily understandable by linguists. This will help the teacher to restructure the examples to formulate more logical rules and segments.

5 Results and Error Analysis

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 grammatical features, our system is able to further segment the affixes into possible and valid prefixes and suffixes. With 140 training examples containing words, the stem and codified morphological features, the system is able to learn and extract 6 prefixes and 25 suffixes. Moreover, the system has learned 70 stem affix extraction rules. As stated in the experiment section, one major limitation of the approach is that the system is not able to look back on previously acquired affixes and do further segmentation.

```
stem(Word, Stem, [2, 1, 1, 2]):-
        set_affix(Word, Stem, [te], [], [[ku],[lh]], []),
        feature([2, 1,1, 2], [passive, perfective, fpsn, spsm, pos]),
        template(Stem, [1, 0, 1, 0, 1]).

stem(Word, Stem, [2, 1, 1, 2]):-
        set_affix(Word, Stem, [[al],[te]], [], [[ku],[lh],[m]], []),
        feature([2, 1,1, 2], [passive, perfective, fpsn, spsm, neg]),
        template(Stem, [1, 0, 1, 0, 1]).
```

FIGURE 10 – New Rules with Affix Segmentation Result

Another main advantage of such learned rules is the ability to review the rules and be verified by a linguist for correctness. This benefit can also be used by the teacher to decide on how to order the examples for better knowledge acquisition. If a certain example presentation generates wrong or incorrect segmentation, the teacher can easily rearrange the list of examples and see a corrected or better segmented result. Such experiments by the example provider could also help to identify complexity of word structure which is taken to be trivial for experts.

An experiment was also done to see the effect of order of examples has on the segment learning predicate. The attempt showed that most of the errors in the segmentation process arise from the ordering of examples. Thus, for example, if the system encounters the affix *[kulh]* before *[ku]*, then, the system puts *[kulh]* in its database of suffixes rather than segmenting it into *[[ku],[lh]]*. With the same token for the suffixes *[ku]* and *[k]* which are subject first person and second person masculine markers, the order of presentation might confuse the learner. If the system encounters a word with the suffix *[k]* first, then the second suffix will be spuriously segmented into *[[k], [u]]* .This necessitates that example presentation to the system should be done in an incremental way by an intelligent teacher. One of the limitations of the system, as explained above, is lack of correcting previously acquired knowledge to reformulate such rules.

In Amharic, as in many other languages with multiple affixes, the affixes may change their form in particular phonological or orthographic environments. In finite-state morphology, these changes are captured in alternation rules. Although our ILP system succeeds in learning some of the alternation rules that play a role in root-template combination and at the boundaries between stems and prefixes or suffixes, we have not yet incorporated the learning of alternation rules into the component of the system that learns to segment prefixes and suffixes through incremental presentation of examples. In the future, we will experiment with the possibility of taking advantage of the teacher's knowledge of the alternate forms of an affix to learn from successive presentations of the same affix in different environments.

The other limitation of 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 valid morpheme combinations, CLOG is inadequate because it is not able to use variables as part of the body of the predicates to be learned.

To measure the effectiveness of the system, precision and recall is used to see the ratio of valid segmentation that exists in the data with the segmentation done by the system. From the 140 words provided to the system, a linguist extracted 221 valid segmentations. It should be noted here that some of the segmentations might appear in a number of instances in the data. From the same data set the system is able to extract 227 segmentations while 215 of this segmentation are correct segmentations that match with the linguist's analysis. The system has shown over generation of segments. The following figure shows the precision and recall values according to the statistics presented above.

$$Precision = \frac{Number\ of\ Correct\ Segmentation}{Total\ Number\ of\ Segmentations\ Found} = \frac{215}{227} = 0.94$$

$$Recall = \frac{Number\ of\ Correct\ Segmentation}{Total\ Number\ of\ Segmentations\ in\ the\ Data} = \frac{215}{221} = 0.97$$

FIGURE 11 – Precision and Recall result

The precision and recall result is satisfactory enough to implement the system in large scale examples and words with more complexity.

We are currently implementing the concept of mutual information to build knowledge from already known facts and rules entertaining partial information. The concept of partial information could be integrated with ILP to generate more rules from rules handling features and affix co-occurrences not found in the training example.

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 experiment also showed that affixes could further be segmented into possible valid prefixes and suffixes during the learning process by building knowledge of affixes on the fly using incremental learning methods. Our previous implementation have gone 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. The current update aiming on affix manipulation also succeeds in segmenting affixes into valid prefixes and suffixes using database generated on the fly during the training phase. As the rules and segmentations are presented in easy and human understandable list of rules, the teacher could restructure the examples and feed the learner to gain better result as needed.

References

Altenbek, G. (2009). *Kazakh Segmentation System of Inflectional Affixes.* Xinjiang University, Harbin, China: Natural Science Foundation of China

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.

Cavar D, Rodrigues P. and Schrementi G. (2005), *Using Morphological and Distributional Cues for Inductive Part-of-Speech Tagging*, CiteSeer.

Chomsky, Noam. (1981). *Lectures on government and binding.* Dordrecht, Dordrecht Foris

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.

Finley Sara (2012), *Morpheme Segmentation in School-Aged Children.* In A. Fine (Ed.) University of Rochester Working Papers in the Language Science Vol 6.

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.

Mulugeta, W and Gasser, M. (2012). *Learning morphological rules for Amharic verbs using inductive logic programming*. SALTMIL-AfLaT Workshop on Language Technology for Normalisation of Less-Resourced Languages. Istanbul.

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

Pirrelli V and Herreros I, (2007), *Learning Morphology by Itself*, On-line Proceedings of the Fifth Mediterranean Morphology Meeting (MMM5). Fréjus 15-18 , University of Bologna.

Rens Bod, Remko Scha (1996) *Data-Oriented Language Processing: An Overview*. ILLC Technical Report LP-96-13, Department of Computational Linguistics, University of Amsterdam

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

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.

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