

L³MORPHO 0.9 User's Guide

Michael Gasser
Indiana University, School of Informatics and Computing
gasser@cs.indiana.edu
28 September, 2009

1. Introduction

L³MORPHO is a Python program that analyzes and generates morphologically complex words in several languages. L³MORPHO uses finite state transducers augmented with grammatical constraints in the form of feature structure descriptions. For more on finite state morphology, see Beesley and Karttunen (2003); for a description of the specific approach used in L³MORPHO, see Gasser (2009).

In the rest of this document, it is assumed that you have at least a basic knowledge of the language(s) you will be working with, of basic grammatical terminology, and of Python.

2. Installation

If you haven't already done it, uncompress the file that you downloaded. This will yield a directory called `L3Morpho-0.9`, which contains all of the files that you need to run L³MORPHO.

You don't have to install the program to use it. In fact since the data for each of the languages will be changing considerably in the coming months, you may want to wait and just put the directory in a handy place and run the program from there.

To install L³MORPHO, change to the `L3Morpho-0.9` directory, and do the following.

```
(1) python setup.py install
```

3. Use

a. STARTING THE PROGRAM

If you have installed L³MORPHO, you start the program by doing the following.

```
(2) >>> import l3
```

If you have not installed it, change to the `L3Morpho-0.9` directory and run the `import` statement above. If this fails, you probably don't have the current directory in your search path. To fix this, do the following.

```
(3) >>> import sys; sys.path.append('')
```

Now the `import` statement should work.

L³MORPHO has three kinds of functions, for analyzing individual words, for analyzing all of the words in a file, and for generating words.

b. ANALYZING A WORD

To analyze a single word, enter the following.

```
(4)  l3.anal_word(language, word)
```

where *language* is an identifier for one of the supported languages and *word* is a word in the form of a Python string. Supported languages and their identifiers include Amharic ('Amharic', 'am'), Tigrinya ('Tigrinya', 'ti'), Spanish ('Español', 'es'), K'iche' ("K'iche'", 'ki'), and Quechua ('Quechua', 'qu').

The program first loads the morphological data for the language if this has not already happened. It then attempts to analyze the word using the built-in finite state transducers that encode lexical and grammatical information about words. Note that the program only has knowledge about certain categories of words: nouns and verbs for Amharic, verbs only for the other languages. If the program can analyze the word, it prints out the root or stem and a description of the word's grammatical features. Here are examples for each of the supported languages. Note that in the case of ambiguity, all analyses are printed out.

```
(5)  >>> l3.anal_word('am', 'ለዘመዶቻችንም')
Loading morphological data for Amharic ...
Word: ለዘመዶቻችንም
POS: noun, stem: ዘመድ
    possessor: 1, plur
    grammar: plural
    preposition: le, conjunctive suffix: m

>>> l3.anal_word('ti', 'ምደለኻዮ')
Loading morphological data for Tigrinya ...
Word: ምደለኻዮ
POS: verb, root: <dly>, citation: ደለየ
    subject: 2, sing, masc
    object: 3, sing, masc
    grammar: perfective
    conjunctive prefix: mI

>>> l3.anal_word('es', 'volvieron')
Cargando datos morfológicos para español ...
Palabra: volvieron
CL: verbo, raíz: <volver>
    pretérito; sujeto: 3, plur

>>> l3.anal_word('qu', 'munawankichis')
Cargando datos morfológicos para quechua ...
Palabra: munawankichis
CL: verbo, raíz: <muna>
```


d. GENERATING A WORD

To generate words using L³MORPHO, you will need to be familiar with the way roots, stems, and grammatical features are represented in the program. With no grammatical features provided, the grammatical structure of the word that is generated conforms to a default set of features for the part-of-speech of the word. For Spanish and Quechua verbs, this is the third person singular present (with no object suffix in Quechua); for Amharic and Tigrinya verbs, it is the third person singular masculine perfective with no object suffix; for K'iche' verbs, it is the incompletive with third person singular absolutive and no ergative prefix or category (termination) suffix; for Amharic nouns, it is the noun stem with no affixes.

To generate a word in its default form, enter the following.

```
(10) >>> 13.gen(language, part_of_speech, root_stem)
```

where *language* is an identifier for one of the supported languages, *part_of_speech* is either 'v' or 'n', and *root_stem* is a string representing the root or stem of a word. Here are examples from all of the languages. (The representation of verb roots for Amharic and Tigrinya is beyond the scope of the document.)

```
(11) >>> 13.gen('am', 'n', 'meng^st')
መንግሥት
>>> 13.gen('ti', 'v', "`d_g")
ዐደገ
>>> 13.gen('es', 'v', 'aprobar')
aprueba
>>> 13.gen('qu', 'v', 'piska')
piskan
>>> 13.gen('ki', 'v', 'war')
kwar
```

To specify grammatical structure other than the default, you put this after the root or stem. It takes the form of a feature structure, consisting of feature-value pairs. Each feature-value pair that you include represents a change in the default set of features. Feature structures are represented as strings of bracketed expressions. A simple feature-value pair is represented by the feature and the value separated by an equals sign. For example, if you want to make an Amharic or Tigrinya verb passive, with no other features in the default changed, you would set the *voice* feature (*vc*) to *passive* (*ps*).

```
(12) >>> 13.gen('ti', 'v', "`d_g", '[vc=ps]')
ተዐደገ
```

Values of `True` and `False` can be represented using + or - signs before the feature name. For example, to make a Quechua verb negative (*neg*), you would do the following.

```
(13) >>> 13.gen('qu', 'v', 'piska', '[+neg]')
piskanchu
```

To specify the subject or object of a verb (the absolutive or ergative argument for K'iche'), you will need a nested feature structure because the value of the subject and object features is itself a feature

structure. For example, to make the subject (sb) of a Quechua verb first person (p1) singular and the object (ob) second person (p2) plural (p1), you would do the following.

```
(14) >>> l3.gen('qu', 'v', 'piska', '[sb=[+p1],ob=[+p2,+p1]]')
piskaykichis
```

For Amharic, you can generate nouns that are derived from verbs. In this case, you specify a verb root and with the feature structure, you specify which category of deverbal noun you want as the value of the *v* feature: infinitive (*inf*), agentive noun (*agt*), instrumental noun (*ins*), or manner noun (*man*).

```
(15) >>> l3.gen('am', 'n', 'sdb', '[v=agt]')
ሰዳቢ
```

The features and values that make up the description of the grammatical structure of a word are based on a particular theory of the morphology of the language in question. These theories are not discussed in this document. However, it is possible to get an idea of how the structure is represented by looking at the features and values that are selected from. To see the features and their possible values for a given language and part-of-speech, do the following.

```
(16) >>> l3.get_features(language, part_of_speech)
```

where *language* is an identifier for one of the supported languages and *part_of_speech* is either 'v' or 'n'. The function *get_features* returns features and values as a Python dict. Here's what gets returned for Spanish verbs.

```
(17) >>> l3.get_features('es', 'v')
{u'dob': {u'p2': [True, False], u'anim': [True, False], u'xpl': [True, False], u'fem': [True, False], u'pl': [True, False], u'fam': [True, False], u'pl': [True, False]}, 'pos': ['v', u'adj', u'adv', u'n'], u'as': {u'cnt': [True, False], u'prf': [True, False]}, u'refl': [True, False], u'sb': {u'p2': [True, False], u'pl': [True, False], u'pl': [True, False], u'fam': [True, False]}, u'tam': {u'prt': [True, False], u'fut': [True, False], u'prf': [True, False], u'ipf': [True, False], u'sub': [True, False], u'tns': [True, False], u'cnd': [True, False], u'imv': [True, False], u'cnt': [True, False]}, u'iob': {u'p2': [True, False], u'xpl': [True, False], u'pl': [True, False], u'pl': [True, False], u'fam': [True, False]}}
```

References

- Beesley, K.R., and Karttunen, L. (2003). *Finite state morphology*. Stanford, California: CSLI Publications.
- Gasser, M. (2009). Semitic morphological analysis and generation using finite state transducers with feature structures. *Conference of the European Chapter of the Association for Computational Linguistics, 12*.