# Morphology and Python

Steven Butler (srbutler@gmail.com)

April 28th, 2016

## Overview

The goal of this talk is to introduce:

- Some basic concepts about morphology, a sub-discipline of linguistics
- Morfessor, a morphological segmentation library in Python
- My work, which tries to use Morfessor to work with patterns it doesn't like very much

## NLP and Segmentation: why bother?

Morphological segmentation is not glamorous. However, it can be an important part of the tool-chain for other types of NLP work, including:

- keyword search
- machine translation
- speech recognition

Segmentation tools that can work with **low-resource languages** are crucial for ensuring that NLP-related services are made available to people who speak languages besides the small number of well-resourced languages (English, German, Chinese, French, etc.).

## But first...

What's a word? *Who knows!*

Words can be defined in a number of ways, including (non-exhaustively)

- **phonologically**: sound, stress, and prosodic patterns
- **syntactically**: word order constraints and constituency
- **orthographically**: writing system, including traditions of word-division

For the purposes of text-based NLP work, **orthographically** almost always wins out.

## Morphology: what’s in a word?

Morphology is:

- "the science of form" (OED)
- the study of the internal structure of words (Haspelmath & Sims, 2010)
- the subdiscipline of linguistics that people usually mean when they say "that language has a lot of grammar" (me)

A morpheme is "the smallest meaningful part of a linguistic expression that can be identified by segmentation". Morphemes can be **bound** or **free** (and anywhere in-between).

## Morphology in English

English is not known for the complexity of its morphology:

```
    words ⟶ word -s

    presented ⟶ present -ed

    undecomposable ⟶ un- de- compos(e) -able
```

Of course, it can get a little tricky sometimes:

```
    feet ⟶ foot + ???
```

## Morphology in Other Languages

Many other languages throughout the world use more complex patterns quite often. These patterns are more productive in these languages:

Spanish:

*desafortunadamente* "unfortunately" ⟶ des- a- fortunada -mente

Indonesian:

*memperkenalkan* "to introduce" ⟶ meN- per- kenal -kan

Turkish:

*uygarlaştıramadıklarımızdanmışsınızcasına*
"behaving as if you are among those whom we could not civilize" ⟶ *another day*...

## Concatenation

What's key to understanding most of these patterns is that they're, broadly, **concatenative**. They involve, more or less, the repeated addition of affixes to a base.

Other patterns found commonly in languages throughout the world, like compounding, are also concatenative

```
    database ⟶ data + base
```

Concatenative patterns are a manageable problem in NLP, and that's what Morfessor handles pretty well.

## Beyond Concatenation

Not all patterns are so easy to deal with, of course. Some patterns are non-concatenative.

Arabic triliteral roots:

*kataba* "he wrote" ⟶ k-t-b (root) + a-a-a  
*taktubu* "she writes" ⟶ ta + k-t-b (root) + u-u  
*kitaab* "book" ⟶ k-t-b (root) + i-aa

These patterns can barely be described using the same tools as concatenative morphological patterns, much less segmented in the same way. The roots have a **discontinuity** that makes matching them with other words from the same root more complex.

## Patterns in Tagalog

Non-concatenative patterns come in many forms. A particular group of patterns got me interested in working with Tagalog:

***basa*** "read" verbal paradigm (drastically simplified):

| **Focus Type** | **Perfective** | **Imperfective** | **Contemplative** |
|---|---|---|---|
| *actor* | bumasa | bumabasa | babasa |
| *object* | binasa | binabasa | babasahin |

# Patterns in Tagalog

These patterns make use of infixation and (partial) reduplication:

**infix**: an affix that attaches inside a root

```
    basa + -in-    ⟶    b -in- asa    ⟶    binasa
```

**reduplication**: the repetition of all or part of a base

```
    basa + <redup>    ⟶    ba- basa    ⟶    babasa
```

With their powers combined...

```
basa + <redup> + -in-  ⟶  ba-basa + -in-  ⟶  b -in- abasa  ⟶  binabasa
```

## Morfessor

Morfessor is a family of algorithms originally introduced in 2005. A widely used implementation is Morfessor 2.0, an open-source Python library. It is widely-used as both a baseline for other segmenters, and as a tool for researchers who need an easy-to-use, language-agnostic segmenter quickly.

Morfessor is **unsupervised**--it is trained on an unannotated corpus. It performs a "greedy and local" search through each word in the corpus, trying to find a the optimal segmentation for the word in the context of all of the other segmentations proposed for the corpus. It stores this information as a **lexicon** and **grammar**.

## My Work

Morfessor is good at finding concatenative morphology.

Morfessor is extraordinarily bad at finding non-concatenative morphology.

I wanted to test whether one can make a concatenative segmenter like Morfessor work with non-concatenative data, like in Tagalog. If Morfessor can be supplemented with tools for other types of morphology, it might work better for more languages.

My basic idea: **Supervise it** (a little)

## Infixer

I set up a system to use regular expressions to pre-process and reshape the data so that known difficult patterns can be accounted for.

> **Search:** `r'^(?P<CON>\w)(um)(?P<VOW>\w)((?P=CON)(?P=VOW)\w*)'`
> **Rewrite:** `r'<redup>-<\g<2>>-\g<4>'`

The regular expressions **linearize** their target patterns to make them amenable to a concatenative analysis:

```
bumabasa ⟶ <redup>-<um>-basa
```

This sidesteps the issues with Morfessor’s search process, but also sacrifices the advantages of completely unsupervised segmentation.

## Other Tools for Segmentation

The Natural Language Toolkit (NLTK) is the natural recommendation. It has specialized tools for working with morphology in greater detail than something like Morfessor.

Tools provided by NLTK and other libraries (like Standford's CoreNLP) include things like stemmers and lemmatizers. These tools can find and organize words around their stem or root forms. When available, use them! They're going to be more accurate.

# Bibliography

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python* (1st ed.). Sepastopol, CA: O'Reilly Media, Inc.

Creutz, M., & Lagus, K. (2005). Inducing the Morphological Lexicon of a Natural Language from Unannotated Text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)* (pp. 106–113).

Haspelmath, M., & Sims, A. D. (2010). *Understanding Morphology* (2nd ed.). London, UK: Hodder Education.

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing Computational Linguistics and Speech Recognition*.

Lewis, M. P., Simons, G. F., & Fennig, C. D. (2016). *Ethnologue: Languages of the World, Nineteenth edition*. (M. P. Lewis, G. F. Simons, & C. D. Fennig, Eds.) (19th ed.). Dallas, Texas: SIL International. Retrieved from http://www.ethnologue.com.

Manning, C. D., Bauer, J., Finkel, J., Bethard, S. J., Surdeanu, M., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. Retrieved from http://aclweb.org/anthology/P14-5010

Schachter, P., & Otanes, F. T. (1972). *Tagalog Reference Grammar*. Berkeley, CA: University of California Press.

Virpioja, S., Smit, P., Grönroos, S.-A., & Kurimo, M. (2013). *Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline*. Retrieved from http://urn.fi/URN:ISBN:978-952-60-5501-5