Statistical morpheme segmentation in Inuktitut

Jessica Huynh

December 19, 2016

1 Introduction

Inuktitut is an Inuit language spoken mostly in Canada, and like many other Eskimo-Aleut languages, it is highly agglutinative, although not very fusional.[6]

2 Background

The highly regular nature of Inuktitut morphology means that one can write a rule-based morpheme segmenter with relatively high accuracy. Indeed, the National Research Council of Canada has already built one, claiming over 95% accuracy on the Nunavut Hansard corpus and similar accuracy on Inuktitut web pages, composing a list of thousands of roots and suffixes to do so.[5]

3 Experiment

3.1 Corpora

For the main corpus, I used the Inuktitut language portions of the Nunavut Hansard.[4] The data in the hansard is formatted as such:

```
**************
qinuvvigijumavakka pigiaqtitsiqqaujuq aippiiqqaujurlu pigiaqtitauqqaujumut tasiqsiqullugit
uqaqtimik uqaqtiup iksivautangannut.
-----
I would ask that the mover and the seconder of the motion escort the Honourable Speaker to
the Speaker's chair.
**************
```

As such, it was a relatively simple task to extract just the Inuktitut portions. I opted to only take the first 25,000 lines or so, to keep the size manageable.

For a secondary corpus, I used portions of the Inuktitut Bible, which was completely translated by the Canadian Bible Society in 2012.[7] The text is in syllabics, so to make it easier to work with, I used the National Research Council of Canada's Inuktitut transcoder to get a Latin alphabet transcription. For those characters it did not successfully transliterate (namely, those oriented facing upwards and with a dot for a long vowel), I used a syllabary resource[6] and did the substitution in a text editor myself. The syllabics that needed manual substitution are in `syllabics.txt`.

3.2 Tools

3.2.1 Morfessor

Morfessor[8] is a package of machine learning methods designed for segmentation tasks, provided as both command-line and a Python package.

Morfessor considers morphemes separate from *morphs*, which could be considered like the surface representation of a morpheme, being the word segments in the data. At its core, Morfessor treats each word as a Hidden Markov Model, with the morphs as a sequence of observations. The hidden states are categories of morphs.[2]

3.2.2 Morphological analyzer

The National Research Council of Canada (NRC) has the Uqailaut project, a rule-based morphological analyzer for Inuktitut.[5] It is provided as a Java .jar file to download and in this project provides the 'ground truth' for the test sets, as well as the annotated data used to bootstrap some of the models.

While there are supposed to be some Java methods packaged within the .jar that can be incorporated into other programs, in practice I couldn't get this to work and instead just grabbed the output of the .jar for each word I wanted the morpheme breakdown for.[1]

3.2.3 Transcoder

The NRC also has an Inuktitut transcoder, to transcode between Inuktitut syllabics and the Roman alphabet.[3]

3.3 Method

The GitHub repository for this project is at `https://www.github.com/jessicahuynh/inuktitut-morpheme-segment`. Broadly, there is a pipeline script, `pipeline.sh`, that prepares the data and models and runs the different experiments before evaluating them.

3.3.1 Creating test and train datasets

The file `scripts/Corpus.java` contains all the code to take in the various corpora, put them into the correct format to be read by Morfessor, and split them into training and test sets at about a 4:1 ratio.

3.3.2 Modeling

This part of the pipeline is relatively straightforward. Morfessor provides command line tools to take in a corpus and build a binary to be read in later for the model.

3.3.3 Training, decoding, and evaluation

The script `scripts/run_hmm.py` does the training, decoding, and evaluation, all using Morfessor's built-in methods.

3.4 Approaches

Morfessor's primary mode of operation is to do unsupervised training, but I also opted to use its built-in methods to do some semi-supervised training with an additional annotated corpus (that is, one where the morpheme boundaries are already given) and do some manual adjusting of the weighting of the annotated corpus.

4 Results

5 Discussion

6 Conclusion

References

[1] java runtime.getruntime() getting output from executing a command line program. http://stackoverflow.com/questions/5711084/java-runtime-getruntime-getting-output-from-executing-a-command-line-program.

[2] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning.

[3] National Research Council of Canada. Inuktitut transcoder. http://www.inuktitutcomputing.ca/Transcoder/index.php?la

[4] National Research Council of Canada. The nunavut hansard: Inuktitut-english parallel corpus. http://www.inuktitutcomputing.ca/NunavutHansard/info.php?lang=en, 2008.

[5] National Research Council of Canada. The uqailaut project: Inuktitut morphological analyzer. http://www.inuktitutcomputing.ca/Uqailaut/info.php, 2012.

[6] Omniglot. Inuktitut @ omniglot. http://www.omniglot.com/writing/inuktitut.htm.

[7] Canadian Bible Society. Eastern arctic inuktitut bible. https://www.bible.com/versions/455-eaib-aclaait-ityorngnaaittot.

[8] Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. *Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline.*