Tech Review: Japanese NLP Analysis with Stanza

W. Daniel Child

1. Overview

Stanza is a Python-based universal natural language analysis package from Stanford that claims to provide a full-fledged NLP analysis pipeline for multiple languages. Since I am interested in NLP work specifically related to Japanese, I wanted to see how it would perform with that language specifically.

2. Setup

Stanza claims ease of implementation and setup, the ability to perform tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) and morphological features tagging, dependency parsing, and named entity recognition.

• Installation

Theoretically, installation is simple enough using pip.

```
>>> pip install stanza
```

Although Stanza can be installed using Anaconda for Python 3.7 or earlier, conda installs will not work if using Python 3.8. (I inadvertently tried running stanza from Python 3.7 within conda after a pip install and promptly received missing resource errors. So if you use Anaconda regularly and want to use Stanza, you should install it as follows:

```
>>> conda install -c stanfordnlp stanza
```

Taking this step promptly resolved error messages involving missing resources.

• Basic Use

Once inside Python, you can import stanza into Python as you would using any standard library. In the examples below, again I assume you want to work with Japanese.

```
>>> import stanza
```

downloads the basic stanza package, but you will be unable to run any commands until you download the language model for the language you plan to use. In our case, that means

```
>>> stanza.download('ja')
```

You would use 'en' if you wanted to experiment with English instead of Japanese. Now that the basic package is installed you need to create an NLP processing pipeline, like so:

```
>>> nlp = stanza.Pipeline('ja')
```

This step was also successful, producing the NLP processing modules one would expect to use.

```
2020-11-14 15:44:55 INFO: Loading these models for language: ja (Japanese):
=======================
| Processor | Package |
-----------------------
| tokenize  | gsd     |
| pos       | gsd     |
| lemma     | gsd     |
| depparse  | gsd     |
=======================

2020-11-14 15:44:55 INFO: Use device: cpu
2020-11-14 15:44:55 INFO: Loading: tokenize
2020-11-14 15:44:55 INFO: Loading: pos
2020-11-14 15:44:56 INFO: Loading: lemma
2020-11-14 15:44:56 INFO: Loading: depparse
2020-11-14 15:44:57 INFO: Done loading processors!
```

You can now start processing Japanese documents. In this review, we'll simply try a few simple sentences to see what Stanza spits out.

I was curious what would happen if English words were included, I copied a definition of NLP written in Japanese.

```
>>> text1 = 'NLPとは、Neuro Linguistic Programing（神経言語プログラミング）の頭文字から名付けられています。'
>>> doc = nlp(text1)
```

Curiously, the first time I ran this, this led to an error, because the English words in this Japanese sentence were unsupported. However, when I reran this code after installing the English processing modules, then Stanza was able to parse the code. The full parsing is rather long, but if you want to see how many words were parsed, you could simply type:

```
>>> doc.num_words
20
```

Next, I tried parsing ‘kuromoji’ (黒文字) which literally means “black text” but which is also the name of a Japanese tokenizer.

```
text2 = '黒文字' #kuro ‘black’ moji ‘characters’
```

This text was incorrectly parsed as follows:

```
[
  [
    {
      "id": 1,
      "text": "黒文",
      "lemma": "黒文",
      "upos": "PROPN",
      "xpos": "NNP",
      "feats": "Number=Sing",
      "head": 0,
      "deprel": "root",
      "misc": "start_char=0|end_char=2",
      "ner": "O"
    },
    {
      "id": 2,
      "text": "字",
      "lemma": "字",
      "upos": "PROPN",
      "xpos": "NNP",
      "feats": "Number=Sing",
      "head": 1,
      "deprel": "flat",
      "misc": "start_char=2|end_char=3",
      "ner": "O"
    }
  ]
]
```

As you can see the first two characters were treated as one word, and the final character as a separate word. It is true that technically one could think of interpret 黒文 as a word ('black writing', probably pronounced 'heibun'), here it is clear that 文字 (characters 'moji') should be together, and the entire phrase should be parsed as black 黒 'kuro' characters 文字 'moji'. Granted, '黒文字' is not a very common expression, but the fact that this was mis=parsed is troubling. This error was repeatedly produced the unambiguous corollary "red characters." (赤文字 akamoji). Some proper names were correctly parsed (e.g. Kansai Airport), but I think these are expected proper nouns, and are obviously built into the dictionary.

In any case, once you have parsed a document, you can *theoretically* have access various properties such as text, sentences, entities, num_tokens, and num_words. I say "theoretically" because when I typed in a simple sentence "そのようなもの（は）食べたくない。" (I don't want to eat that kind of thing.) For some reason, the first time I ran this code, the results were extremely disappointing. The text was parsed as a single word!

```
>>> text3 = 'そのようなもの食べたくない。'
>>> doc3 = nlp(text3)
```

```
[
  [
    {
      "id": 1,
      "text": "そのようなもの食べたくない。",
      "lemma": "そのようなもの食べたくない。",
      "upos": "NOUN",
      "xpos": "NN",
      "feats": "Number=Sing",
      "head": 0,
      "deprel": "root",
      "misc": "start_char=0|end_char=14",
      "ner": "O"
    }
  ]
]
```

However, after playing around with the imports and configuration and rerunning this code, I was able to get correct parsing (too long to show here).

• Additional Information

Once you have parsed the code, there are various kinds of information you can look into. The parsed objects include Document, Sentence, Token, Word, and Span, and each of these enable you to drill down for more information.

Although Stanza has a wide variety of features, many of these are not available for Japanese. Specifically, the named entity recognition, sentiment analysis, and biomedical models are not available.

4. Conclusion

Overall, Stanza appears to be a promising, though limited, avenue for Japanese text parsing. It does seem to be able to handle the basics, even though some morphological analysis seems a bit simplistic and probably benefit from a bigger dictionary. Once one navigates past the conda / non-conda installation issues, it appears to work fairly well, though the fact that different parse results were generated on different attempts is puzzling.

5. References

There are no direct references on Stanza for Japanese (my interest and the focus of this review), but general Stanza information can be found at:

https://stanfordnlp.github.io/stanza/