Alexander Scharf
CS410 (Fall 2020)
Tech Review

Japanese NLP with SudachiPy, spaCy, and GiNZA

Intro

NLP in Japanese language is mature and effective, yet information is not necessarily available in a centralized location, particularly in English. This tech review will focus on Japanese NLP with the spaCy framework, highlighting particular challenges between Western languages and recent trends.

A major difference between Western languages and Japanese is the lack of spaces. Hence, tokenization, a relatively trivial task in Western NLP, is quite a difficult challenge in Japanese. Since tokenization cannot be done based upon spaces, in Japanese it is typically done together with parts-of-speech tagging.

Morphological analysis

One of the most well-known and widely-used morphological analyzers is MeCab[1]. While MeCab is well-proven, it has not been updated in nearly seven years and does not have close integration with other NLP tools. The morphological analyzer that we'll be looking at for this review and utilized by spaCy is SudachiPy[2], the Python version of Sudachi.

SudachiPy additionally requires a dictionary file. Three different sizes of dictionaries[3] for Sudachi. Since Japanese does not have spaces and some words in Japanese are compounds of other words, Sudachi has three modes for splitting words: A, B, C, where in mode A texts are divided into the shortest units, C into named entities, and B into middle units:

```
mode = tokenizer.Tokenizer.SplitMode.C
print( [m.surface() for m in tokenizer_obj.tokenize( sample, mode)] )
# ['国家公務員']

mode = tokenizer.Tokenizer.SplitMode.B
print( [m.surface() for m in tokenizer_obj.tokenize(sample, mode)])
#['国家', '公務員']

mode = tokenizer.Tokenizer.SplitMode.A
print( [m.surface() for m in tokenizer_obj.tokenize(sample, mode)])
#['国家', '公務', '員']
```

[1] https://taku910.github.io/mecab/
[2] https://github.com/WorksApplications/SudachiPy
[3] https://github.com/WorksApplications/SudachiDict

In addition to spacing, another challenge of Japanese is that there are often words that have variations, frequently from using one of Japanese's three different scripts (e.g. かつ丼 → カツ丼) or variants of kanji characters (e.g. 附属 → 付属). Sudachi can also normalize these forms.

NLP with spaCy

Since version 2.3 released June, 2020, spaCy has had built-in support for Japanese language, including support for SudachiPy and pretrained models.[4] Japanese language works "out-of-the-box," with spaCy, supporting tokenization and parts-of-speech tagging with SudachiPy, a parser, sentenciser, and entity recognizer.

```
import spacy

nlp = spacy.load("ja_core_news_sm")

text = "こんにちは。花子です。はじめまして。"

for w in nlp(text):
    print(f'text:{w.text}, pos:{w.pos_}')
"""
text:こんにちは, pos:INTJ
text:。, pos:PUNCT
text:花子, pos:PROPN
text:です, pos:AUX
text:。, pos:PUNCT
text:はじめ, pos:AUX
text:まし, pos:AUX
text:て, pos:SCONJ
text:。, pos:PUNCT
"""
```

spaCy has three pretrained models - ja_core_news_sm, ja_core_news_md, and ja_core_news_lg, all news/media sources. The medium and above models have word vectors that can be used to compute similarity.[5]

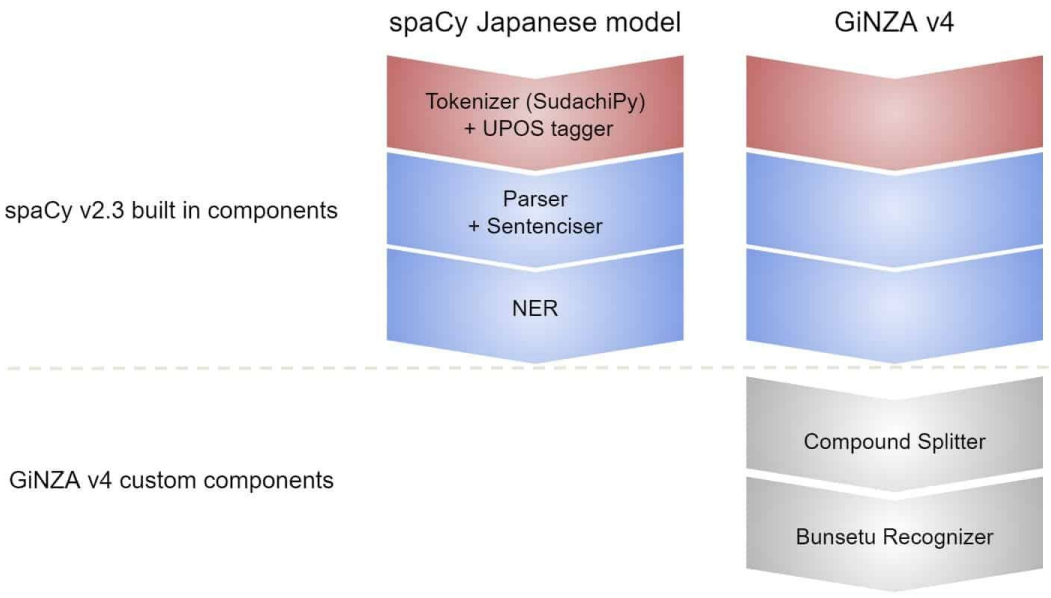
[4] https://spacy.io/usage/v2-3
[5] https://qiita.com/poyo46/items/7a4965455a8a2b2d2971

```python
import spacy

nlp = spacy.load("ja_core_news_md")

text = "大阪は雨が降りました。この週末は雪が降るようです。"

word_dict = {}

for w in nlp(text):
    word_dict[w.text] = w

print(word_dict['雨'].similarity(word_dict['雪']))
#0.6119726
print(word_dict['雨'].similarity(word_dict['大阪']))
#0.17225966
```

In the example above, rain (雨) and snow (雪) had a higher similarity, while rain and Osaka (大阪) had a lower similarity.

Beyond spaCy with GiNZA

Beyond the default components available in spaCy, GiNZA is a Japanese NLP library[6] with some additional functionality, notes in the image below: [7]



[6] https://github.com/megagonlabs/ginza
[7] https://www.megagon.ai/jp/blog/blog-ginza-version-4-0/

Of particular interest is the bunsetu recognizer. Bunsetsu (sometimes romanized as bunsetu) are the smallest possible coherent components of the Japanese language. Because Japanese does not demarcate words with spaces, the concept of a “word” is much different than that of Western languages. Bunsetsu have a meaning-bearing word followed by a string of suffixes, auxiliary verbs and particles to modify its meaning and designate its grammatical role.[8]

```
import spacy
import ginza

nlp = spacy.load("ja_ginza")

doc = nlp("この週末は雪が降るようです。")

print(ginza.bunsetu_spans(doc))
#[この, 週末は, 雪が, 降るようです。]
```

The bunsetu recognizer breaks the sentence down into four syntactic elements each with meaning.

Conclusion

Though facing unique challenges compared with Western languages, Japanese NLP is becoming more accessible and easier to use and implement. Tools to analyze and process Japanese text are now part of standard libraries, while unique functionality for Japanese text is able to be used together with common tools.

[8] https://en.wikipedia.org/wiki/Japanese_grammar