PyCantonese: Cantonese Linguistics and NLP in Python

Jackson L. Lee, Litong Chen[1], Charles Lam[2], Chaak Ming Lau[3], Tsz-Him Tsui
[1]Wheaton College, [2]Hang Seng University of Hong Kong, [3]Education University of Hong Kong

Abstract

This paper introduces PyCantonese, an open-source Python library for Cantonese linguistics and natural language processing. After the library design, implementation, corpus data format, and key datasets included are introduced, the paper provides an overview of the currently implemented functionality: stop words, handling Jyutping romanization, word segmentation, part-of-speech tagging, and parsing Cantonese text.

Keywords: Cantonese, Jyutping, word segmentation, part-of-speech tagging, stop words

1. Introduction

Cantonese is one of the most well-known Chinese varieties other than Mandarin Chinese. To handle Cantonese language data and perform natural language processing (NLP) tasks, there did not exist an open-source software package dedicated to Cantonese. This paper introduces PyCantonese, an open-source Python library for Cantonese linguistics and natural language processing.[1]

PyCantonese prioritizes ease of use in legal terms. It is publicly available for free for all purposes, including commercial ones. All of its dependent software packages, datasets, and models have compatible licenses that allow free and public usage. The drawback is that only a very small number of Cantonese-specific datasets fulfill this requirement. While recent advances in NLP have been largely due to neural network-based machine learning coupled with the availability of a large amount of data, the fact that only a small amount of Cantonese data is legally available to PyCantonese means that it would be unrealistic for PyCantonese to train or include models based on neural networks. Although for this reason state-of-the-art NLP tools are not available, PyCantonese is useful for, among other use cases, creating more annotated Cantonese data, which can then be fed into other NLP systems.

In the following, we first discuss the library design and implementation at a high level (section 2). We then explain the corpus data format adopted by PyCantonese and introduce the datasets included (section 3). This is followed by an overview of the currently implemented functionality: stop words (section 4), Jyutping romanization (section 5), word segmentation (section 6), part-of-speech tagging (section 7), and parsing Cantonese text (section 8).

[1]Excluding the first author, all other authors are listed alphabetically by last name. The version of PyCantonese discussed in this paper is v3.4.0 released in December 2021. URL: `https://pycantonese.org`

2. Design and Implementation

PyCantonese is built with a high level of usability and transparency in mind. The Python programming language is chosen for its popularity in general as well as in computational linguistics and NLP specifically. The software is readily available for download and installation through the Python Package Index (i.e., by running `pip install pycantonese`). Major functionality is exposed as top-level functions for ease of access. The inputs to and outputs from PyCantonese are intuitive Python data structures, for high interoperability with other Python programs.

For transparency, the source code and built-in data of PyCantonese are publicly available on GitHub. The use of Git for version control coupled with the online GitHub interface makes visible the history and development process of the software.

3. Corpus Data: Format and Sources

Many capabilities of PyCantonese, to be introduced in the subsequent sections, would not have been possible without publicly available datasets. This section describes the corpus data format that PyCantonese has adopted and the key datasets currently included in the package.

3.1. CHAT Format

For a corpus to be useful for computational linguistics and NLP work, its source data must be in a well-defined, machine-readable format, and a corresponding parser for this data format must be available. For Cantonese, a focal point of corpora available for research purposes that meet these criteria is the CHILDES database (MacWhinney, 2000), thanks to research on Cantonese language acquisition in recent years. These corpora from CHILDES are conversational data transcribed in the CHAT transcription format with rich linguistic annotations. Given CHAT has been widely used in existing Cantonese corpora for academic research, it is natural for PyCantonese to adopt it as the corpus format. To parse the CHAT format, PyCantonese uses the PyLangAcq package (Lee et al., 2016), a Python library

for handling CHAT conversational data. An immediate benefit of adopting the CHAT data format is that PyCantonese can readily access Cantonese corpora from CHILDES:

```
>>> import pycantonese
>>> url = (
...     "https://childes.talkbank.org/"
...     "data/Biling/YipMatthews.zip"
... )
>>> corpus = pycantonese.read_chat(url)
>>> corpus.n_files()
501
>>> len(corpus.words())
1949480
```

Figure 1: Accessing the Yip-Matthews Cantonese-English bilingual corpus (Yip and Matthews, 2007)

3.2. The Hong Kong Cantonese Corpus (HKCanCor)

The Hong Kong Cantonese Corpus (Luke and Wong, 2015, commonly known as HKCanCor) is a Cantonese corpus based on spontaneous speech and radio programs in Hong Kong from the late 1990's. Its transcribed and released version contains about 200,000 Chinese characters. As of this writing, it is still the only corpus of Cantonese conversation data that is word-segmented, romanized, annotated for parts of speech, has its source data files made publicly available, and is associated with a permissive license for both commercial and non-commercial usage. These properties make HKCanCor an ideal candidate for inclusion in PyCantonese for the purposes of demonstrating how corpus data can be formatted for and accessed through the package, as well as serving as an underlying data resource for much of the functionality of PyCantonese introduced in this paper.

The source data of HKCanCor was in an XML format, and has been converted into the CHAT format. As a sample corpus in PyCantonese, the HKCanCor dataset is readily available (Figure 2).

Through a loaded dataset, PyCantonese provides functions to search for data of interest by criteria such as word forms, pronunciations and part-of-speech tags (Figure 3).

```
>>> import pycantonese
>>> corpus = pycantonese.hkcancor()
>>> result = corpus.search(coda="[ptk]", tone="2")
>>> len(result)
71
>>> result[:5]
[Token(word='雀', pos='N', jyutping='zoek2', ...),
 Token(word='雀', pos='N', jyutping='zoek2', ...),
 Token(word='綠', pos='A', jyutping='luk2', ...),
 Token(word='賊', pos='N', jyutping='caak2', ...),
 Token(word='dut2', pos='O', jyutping='dut2', ...)]
```

Figure 3: Searching for occurrences of *pinjam* "changed tones" with coda -p, -t, and -k in HKCanCor

3.3. rime-cantonese as a Cantonese Lexicon

The rime-cantonese resource (CanCLID, 2021) primarily supports Chinese inputting methods based on the Cantonese Jyutping romanization (more on Jyutping in section 5). For our purposes, rime-cantonese is a Cantonese lexicon with pairs of Chinese or Cantonese characters, on the one hand, and the corresponding romanization in Jyutping, on the other. The source data is regularly updated and expanded, making it a valuable resource to supplement a corpus such as HKCanCor. As of this writing, about 170,000 word-romanization pairs have been incorporated into PyCantonese, for improving results of conversion from Chinese characters to Jyutping romanization (section 5) and word segmentation (section 6).

4. Stop Words

In many NLP tasks, it is often necessary to identify and filter stop words, English examples of which include function words such as pronouns and determiners. Stop words are highly relevant for PyCantonese, as neural network-based NLP approaches, most recent versions of which would not require stop words, are unavailable (see section 1). PyCantonese provides a pre-defined list of approximately 100 Cantonese stop words, which were derived from the most frequent function words in HKCanCor and manually reviewed.

5. Jyutping Romanization

Jyutping is the standard romanization for Cantonese devised by the Linguistic Society of Hong Kong (Tang et al., 2002). PyCantonese provides capabilities for common computational tasks involving Jyutping.

5.1. Characters-to-Jyutping Conversion

PyCantonese provides tooling for converting Chinese characters to Jyutping romanization. Such functionality is powered by the HKCanCor and rime-cantonese data included in the software package.

```
>>> import pycantonese
>>> corpus = pycantonese.hkcancor()
>>> corpus.head()

*XXA:  喂      遲      啲      去       唔      去       旅行            啊      ?
%mor:  E|wai3  A|ci4   U|di1   V|heoi3  D|m4    V|heoi3  VN|leoi5hang4   Y|aa3   ?

*XXA:  你      老公           有冇          平         機票         啊      ?
%mor:  R|nei5  N|lou5gung1    V1|jau5mou5   A|peng4   N|gei1piu3   Y|aa3   ?

*XXB:  平      機票           要       淡季             先       有得           平       嚟       喎       .
%mor:  A|peng4 N|gei1piu3     VU|jiu3  AN|daam6gwai3   D|sin1   VU|jau5dak1   A|peng4  Y|gaa3   Y|wo3    .

*XXB:  而家       旺       -   .
%mor:  T|ji4gaa1  A|wong6  -|  .

*XXA:  冇得           去       嗱       .
%mor:  VU|mou5dak1    V|heoi3  Y|laa4   .
```

Figure 2: Importing the HKCanCor data

```
>>> import pycantonese
>>> pycantonese.characters_to_jyutping("講廣東話")
[('講', 'gong2'), ('廣東話', 'gwong2dung1waa2')]
```

Figure 4: Converting Chinese characters (e.g., "speak Cantonese") into Jyutping

As Figure 4 shows, word segmentation (section 6) is a necessary component of characters-to-Jyutping conversion. This is because a given character may have multiple pronunciations, and that, while still imperfect, word segmentation is an effective strategy for pronunciation disambiguation.

5.2. Parsing Jyutping

It is sometimes necessary to identify the phonological components in Jyutping romanization. For instance, a researcher may be interested in the statistics of Cantonese phonemes in a corpus. Another example is when one is interested in converting Jyutping into another romanization system (section 5.3). Figure 5 shows how PyCantonese can parse Jyutping into component onsets, nuclei, codas, and tones.

```
>>> import pycantonese
>>> # thank you
>>> pycantonese.parse_jyutping("m4goi1")
[Jyutping(onset='', nucleus='m',
          coda='', tone='4'),
 Jyutping(onset='g', nucleus='o',
          coda='i', tone='1')]
```

Figure 5: Parsing Jyutping romanization

5.3. Other Romanization Schemes

Multiple Cantonese romanization systems exist. Apart from the standard Jyutping, another popular one is the Yale system, which is still widely used in Cantonese language teaching materials for its more iconic tone notations using accents. For convenience, Cantonese provides a function for converting Jyutping into the Yale system (Figure 6).

```
>>> import pycantonese
>>> # thank you
>>> pycantonese.jyutping_to_yale('m4goi1')
['m̀h', 'gōi']
```

Figure 6: Jyutping-to-Yale romanization conversion

6. Word Segmentation

Like other Chinese varieties, Cantonese is written in Chinese characters without indication of word boundaries. For this reason, the first task of any computational work that involves Cantonese raw text is word segmentation. PyCantonese provides a word segmenter based on the longest string matching method (see also Fung and Bigi 2015), with a lexicon derived from the HKCanCor and rime-cantonese data (Figure 7).

```
>>> import pycantonese
>>> pycantonese.segment(
...     "廣東話容唔容易學？"
... )  # Is Cantonese easy to learn?
['廣東話', '容', '唔', '容易', '學', '？']
```

Figure 7: Word segmentation

7. Part-of-speech Tagging

Thanks to the fact that the HKCanCor corpus (section 3.2) included in PyCantonese is annotated with part-of-speech tags, PyCantonese comes with a part-of-speech tagger. The tagger uses an averaged perceptron model whose implementation is similar to that in the NLTK package (Bird et al., 2009) and where features are based on word bigrams and trigrams around

a target word. The part-of-speech annotations in the HKCanCor use a tagset of over 100 tags. To facilitate cross-linguistic NLP work, PyCantonese maps a predicted tag to its equivalent from the Universal Dependencies tagset (de Marneffe et al., 2021) with a much smaller tagset of 17 tags (Figure 8).

```
>>> import pycantonese
>>> pycantonese.pos_tag(
...     ['我', '嗰日', '買', '嗰', '對', '鞋', '。']
... )  # I bought that pair of shoes yesterday.
[('我', 'PRON'), ('嗰日', 'ADV'), ('買', 'VERB'),
 ('嗰', 'PRON'), ('對', 'NOUN'), ('鞋', 'NOUN'),
 ('。', 'PUNCT')]
```

Figure 8: Part-of-speech tagging

8. Parsing Cantonese Text

While PyCantonese provides as stand-alone functions the capabilities for word segmentation, part-of-speech tagging, and handling Jyutping romanization, a user interested in analyzing Cantonese text data would like (i) most if not all of these functions to apply to their data, and (ii) a convenient data structure to flexibly access and analyze the parsed data. To this end, PyCantonese offers a high-level function that wraps these currently available NLP functions and outputs a data structure that takes advantage of the PyLangAcq package for handling CHAT-formatted data (Figure 9).

9. Conclusion

This paper introduces the PyCantonese package for Cantonese computational linguistics and NLP work. While the package is under active development, it has become clear that a key constraint for what capabilities are possible is the availability of data. To allow for a broad range of uses, including commercial ones, PyCantonese is associated with a permissive license. For this reason, third-party datasets used by PyCantonese must also have a compatible license. In general, it is uncommon for language resources to be freely available and allow for commercial usage as well, let alone those specifically for Cantonese. As data is a key driver for tools such as PyCantonese, it is hoped that more effort is put in compiling Cantonese datasets and releasing them to the public.

10. Bibliographical References

Bird, S., Loper, E., and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

de Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Fung, R. and Bigi, B. (2015). Automatic word segmentation for spoken cantonese. In *2015 International Conference Oriental COCOSDA held jointly with 2015 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE)*, pages 196–201. IEEE.

Lee, J. L., Burkholder, R., Flinn, G. B., and Coppess, E. R. (2016). Working with CHAT transcripts in Python. Technical Report TR-2016-02, Department of Computer Science, University of Chicago.

MacWhinney, B. (2000). *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Mahwah, NJ, 3rd edition.

Tang, S.-W., Fan, K., Lee, T. H.-T., Lun, C., Luke, K.-K., Tung, P., and Cheung, K.-H. (2002). *Guide to LSHK Cantonese Romanization of Chinese Characters*. Linguistic Society of Hong Kong.

Yip, V. and Matthews, S. (2007). *The Bilingual Child: Early Development and Language Contact*. Cambridge University Press.

11. Language Resource References

CanCLID. (2021). rime-cantonese, `https://github.com/rime/rime-cantonese`.

Luke, K. K. and Wong, M. L. Y. (2015). The Hong Kong Cantonese Corpus: Design and Uses. *Journal of Chinese Linguistics Monograph Series*, 25:312–333.

```
>>> import pycantonese
>>> data = "你食咗飯未呀？食咗喇！你聽日得唔得閒呀？"
>>> # Have you eaten yet? Yes, I have! Are you free tomorrow?
>>> corpus = pycantonese.parse_text(data)
>>> corpus.head()
*X:    你          食           咗          飯           未          呀          ？
%mor:  PRON|nei5   VERB|sik6   PART|zo2   NOUN|faan6   ADV|mei6   PART|aa4   ？

*X:    食          咗          喇          ！
%mor:  VERB|sik6   PART|zo2   PART|laa1   ！

*X:    你          聽日              得          唔          得閒              呀          ？
%mor:  PRON|nei5   ADV|ting1jat6   VERB|dak1   ADV|m4   ADJ|dak1haan4   PART|aa4   ？
```

Figure 9: Parsing Cantonese text