The Journal of Open Source Software

WordTokenizers.jl: Basic tools for tokenizing natural language in Julia

Ayush Kaushal[1], **Lyndon White**[2], **Mike Innes**[3], **and Rohit Kumar**[4]

1 Indian Institute of Technology, Kharagpur **2** The University of Western Australia **3** Julia Computing **4** ABV-Indian Institute of Information Technology and Management Gwalior

DOI: 10.21105/joss.01956

Software

- Review
- Repository
- Archive

Editor: William Rowe

Reviewers:

- @leios
- @ninjin

Submitted: 07 December 2019
Published: 06 February 2020

Summary

WordTokenizers.jl is a tool to help users of the Julia programming language (Bezanson, Edelman, Karpinski, & Shah, 2014), work with natural language. In natural language processing (NLP) tokenization refers to breaking a text up into parts – the tokens. Generally, tokenization refers to breaking a sentence up into words and other tokens such as punctuation. Complementary to word tokenization is *sentence segmentation* or *sentence splitting* (occasionally also called *sentence tokenization*), where a document is broken up into sentences, which can then be tokenized into words. Tokenization and sentence segmentation are some of the most fundamental operations to be performed before applying most NLP or information retrieval algorithms.

WordTokenizers.jl provides a flexible API for defining fast tokenizers and sentence segmentors. Using this API several standard tokenizers and sentence segmenters have been implemented, allowing researchers and practitioners to focus on the higher details of their NLP tasks.

WordTokenizers.jl does not implement significant novel tokenizers or sentence segmenters. Rather, it contains ports/implementations of the well-established and commonly used algorithms. At present, it contains rule-based methods primarily designed for English. Several of the implementations are sourced from the Python NLTK project (Bird & Loper, 2004), (Bird, Klein, & Loper, 2009); although these were in turn sourced from older pre-existing methods.

WordTokenizers.jl uses a `TokenBuffer` API and its various lexers for fast word tokenization. `TokenBuffer` turns the string into a readable stream. A desired set of TokenBuffer lexers are used to read characters from the stream and flush out into an array of tokens. The package provides the following tokenizers made using this API.

- A Tweet Tokenizer (Potts, 2019) for casual text.
- A general purpose NLTK Tokenizer (Bird & Loper, 2004), (Bird et al., 2009).
- An improved version of the multilingual Tok-tok tokenizer (Dehdari, 2015), (Dehdari, 2014).

With various lexers written for the `TokenBuffer` API, users can also create their high-speed custom tokenizers with ease. The package also provides a simple reversible tokenizer (Mielke & Eisner, 2018), (Mielke, 2019), that works by leaving certain merge symbols, as a means to reconstruct tokens into the original string.

WordTokenizers.jl exposes a configurable default interface, which allows the tokenizer and sentence segmenters to be configured globally (where this is used). This allowed for easy benchmarking and comparisons of different methods.

WordTokenizers.jl is currently being used by packages like TextAnalysis.jl, Transformers.jl and CorpusLoaders.jl for tokenizing text.

Other similar software

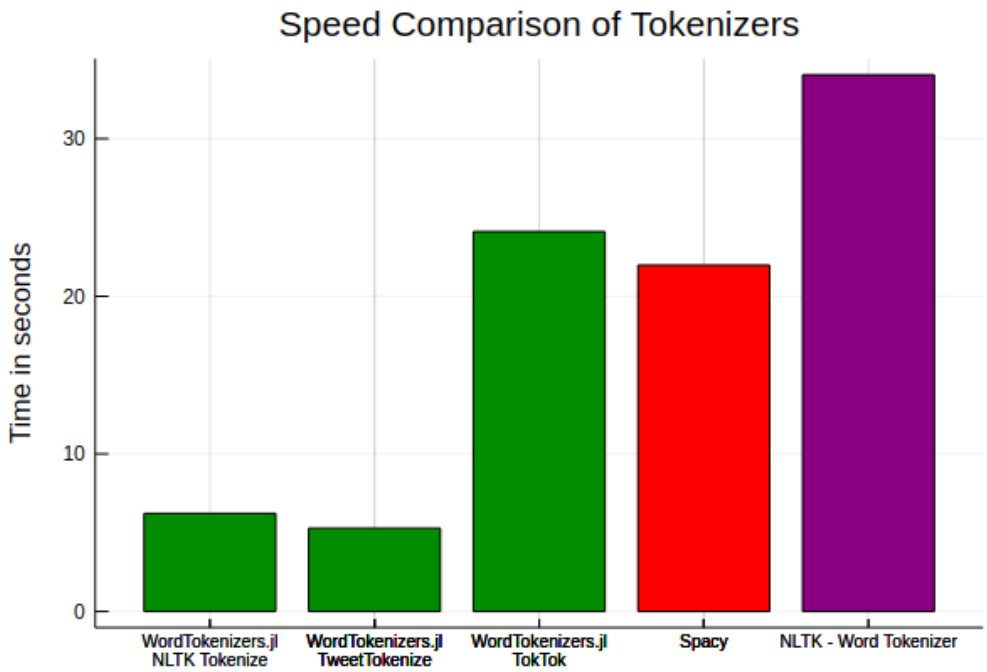

Figure 1: Speed comparison of Tokenizers on IMDB Movie Review Dataset

There are various NLP libraries and toolkits written in other programming languages, available to Julia users for tokenization. NLTK and SpaCy packages provide users with a variety of tokenizers, accessed to Julia users via `PyCall`. Shown above is a performance benchmark of using some of the WordTokenizers.jl tokenizers vs PyCalling the default tokenizers from NLTK and SpaCy. This was evaluated on the ~127,000 sentences of the IMDB Movie Review Dataset. It can be seen that the performance of WordTokenizers.jl is very strong.

There are many more packages like Stanford CoreNLP, AllenNLP providing a couple of basic tokenizers. However, WordTokenizers.jl is faster and simpler to use, providing with a wider variety of tokenizers and a means to build custom tokenizers.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2014). Julia: A fresh approach to numerical computing. doi:10.1137/141000671

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python*. O'Reilly Media, Inc. Retrieved from http://www.nltk.org/

Bird, S., & Loper, E. (2004). NLTK: The natural language toolkit. In *Proceedings of the acl 2004 on interactive poster and demonstration sessions* (p. 31). Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/P04-3031

Dehdari, J. (2014). *A neurophysiologically-inspired statistical language model* (PhD thesis). The Ohio State University.

Dehdari, J. (2015). Tok-tok: A fast, simple, multilingual tokenizer. Retrieved from https://github.com/jonsafari/tok-tok

Mielke, S. J. (2019). A simple, reversible, language-agnostic tokenizer. Retrieved from https://sjmielke.com/papers/tokenize/

Mielke, S. J., & Eisner, J. (2018). Spell once, summon anywhere: A two-level open-vocabulary language model. *CoRR*, *abs/1804.08205*. doi:10.1609/aaai.v33i01.33016843

Potts, C. (2019). Sentiment symposium tutorial: Tokenizing. Retrieved 2011, from http://sentiment.christopherpotts.net/tokenizing.html#sentiment