The Journal of Open Source Software

# Jabberwocky: an ontology-aware toolkit for manipulating text

**Samantha C Pendleton**[1, 2] **and Georgios V Gkoutos**[1, 2]

**1** Institute of Cancer and Genomic Sciences, University of Birmingham, UK **2** University Hospitals Birmingham NHS Foundation Trust, UK

**DOI:** 10.21105/joss.02168

**Software**

- Review
- Repository
- Archive

**Editor:** Mark A. Jensen

**Reviewers:**

- @wdduncan
- @balhoff

**Submitted:** 02 March 2020
**Published:** 01 July 2020



## Summary

Unstructured textual data is underused, as extracting the key textual elements is complicated by a lack of structured terms, e.g., collecting the sentences from a corpus that are discussing a particular topic. To extract valuable text about a topic from a corpus, a user will need to gather a set of related terms. For example, when analysing clinical documents we can extract sentences by using specific clinicial terms. However this can miss additional valuable sentences where synonyms are used instead (e.g., physician notes that use shorthand). By considering terms and their synonyms we can extract more sentences from a corpus, making more data available for analysis. One way to do this and represent our knowledge of terms associated with a domain is to create an ontology. Ontologies allow us to formalise our knowledge of a domain in a condensed manner by using controlled terms, called classes (Hoehndorf, Schofield, & Gkoutos, 2015). Classes can be annotated with metadata, including synonyms. Ontologies can include relationships between terms, and annotations such as cross-references to other ontologies (Hoehndorf et al., 2015).

Clearly, ontologies are valuable for the analysis of textual data. Unfortunately, despite the existence of many well-established ontologies, such as the "Human Phenotype Ontology" (Robinson et al., 2008) and the "Disease Ontology" (Schriml et al., 2012), there remains a lack of tools that can take advantage of ontologies, especially for general text manipulation. Existing tools for annotating text, such as "spaCy" (Honnibal & Montani, 2017), "tagtog" (Cejuela et al., 2014), and "Stanford CoreNLP" (Manning et al., 2014) cannot interrogate text with an ontology directly, and require ontologies to be pre-processed into other formats (leaving the time-consuming task of extracting labels and tags from an ontology into a suitable intermediate format as an exercise for the end-user). These are specialist tools, returning all text in the document with every word tagged, as "noun", "verb", and other customised tags. There exists a niche for users who want to leverage an ontology to retrieve textual data from a corpus without having to perform any pre-processing, or parse away unwanted tags.

We introduce Jabberwocky, a Python-based (Rossum, 1995), open-source toolkit (accessible via https://github.com/sap218/jabberwocky) that allows users to query text in an ontology-aware fashion, and to modify those ontologies based on their findings. For example, with Jabberwocky's `catch` command, a user provides textual data, their chosen ontology, and a set of classes from the ontology to use as search terms. Jabberwocky cleans the input text, collects the annotated synonyms for the user-specified target classes (using "Beautiful Soup" to read the ontology's XML structure (Richardson, 2007)), and then returns the key elements (e.g., lines from a corpus) which match one of the target terms, or a synonym from the ontology. The `catch` command will help users retrieve more matches for their chosen terms from the corpus, without users having to explicitly define all the possible synonyms or alternative spellings beforehand.

Jabberwocky also helps ontology developers to iteratively improve their ontology. The `bite` command allows a user to provide textual data and rank the important terms by using the

term frequency–inverse document frequency (tf-idf) method from "scikit-learn" (Pedregosa et al., 2011), which calculates an importance metric for a term based on the frequency of its occurrence and the document size. Providing an ontology will exclude terms already described in the ontology, meaning the result of `bite` will be a CSV of candidate terms to potentially be added to the ontology, exported by "pandas" (McKinney & Others, 2010). Once an expert has reviewed the terms and associated them to a class in the ontology, Jabberwocky's third command, `arise`, will annotate the classes in the ontology, adding the newly identified synonyms. Iteratively performing multiple rounds of `bite` and `arise` can help the development and maintenance of ontologies. A user could use the `catch` command to confirm the modified ontology now captures more of the corpus.

Jabberwocky's test repository (see Jabberwocky repo for further instructions), shows examples of each command separately. The 'process' directory shows an example that combines all three commands to demonstrate an example workflow. With 24 blog posts, the first use of `catch` returned 11 posts with the provided keywords. The example uses `bite` to review the CSV of ranked terms and curated new synonyms, simply by adding the corresponding class label from the ontology. It then uses `arise` to add the identified synonyms into the ontology. With the second round of `catch` the number of posts returned for the same keywords increased to 16. This is a basic and straightforward example, but powerful. With Jabberwocky, users can efficiently search their text and gain more instances, providing new insight.

Jabberwocky leverages the strength of ontologies and text for a wide range of tasks. It will be useful to users who want to manipulate textual data using controlled vocabulary from ontologies.

## Acknowledgements

Project was funded by the Medical Research Council (MRC) (MR/S502431/1) & supported by Health Data Research (HDR) UK (HDRUK/CFC/01).

## References

Cejuela, J. M., McQuilton, P., Ponting, L., Marygold, S. J., Stefancsik, R., Millburn, G. H., Rost, B., et al. (2014). Tagtog: Interactive and text-mining-assisted annotation of gene mentions in PLOS full-text articles. *Database*, *2014*(0). doi:10.1093/database/bau033

Hoehndorf, R., Schofield, P. N., & Gkoutos, G. V. (2015). The role of ontologies in biological and biomedical research: A functional perspective. *Brief. Bioinform.*, *16*(6), 1069–1080. doi:10.1093/bib/bbv011

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Retrieved from https://github.com/explosion/spaCy

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60). doi:10.3115/v1/p14-5010

McKinney, W., & Others. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56). Austin, TX. doi:10.25080/majora-92bf1922-00a

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, *12*(Oct), 2825–2830.

Richardson, L. (2007). Beautiful soup documentation. *April*. Retrieved from https://beautiful-soup-4.readthedocs.io/en/latest/

Robinson, P. N., Köhler, S., Bauer, S., Seelow, D., Horn, D., & Mundlos, S. (2008). The human phenotype ontology: A tool for annotating and analyzing human hereditary disease. *Am. J. Hum. Genet.*, *83*(5), 610–615. doi:10.1016/j.ajhg.2008.09.017

Rossum, G. van. (1995). Python tutorial, technical report CS-R9526. *Centrum voor Wiskunde en Informatica (CWI), Amsterdam*.

Schriml, L. M., Arze, C., Nadendla, S., Chang, Y.-W. W., Mazaitis, M., Felix, V., Feng, G., et al. (2012). Disease ontology: A backbone for disease semantic integration. *Nucleic Acids Res.*, *40*(Database issue). doi:10.1093/nar/gkr972