Stop Word Lists in Free Open-source Software Packages

Joel Nothman
Sydney Informatics Hub
University of Sydney
joel.nothman@gmail.com

Hanmin Qin
Peking University
qinhanmin2005@sina.com

Roman Yurchak
Symerio
rth.yurchak@gmail.com

Abstract

Open-source software (OSS) packages for natural language processing often include stop word lists. Users may apply them without awareness of their surprising omissions (e.g. *hasn't* but not *hadn't*) and inclusions (e.g. *computer*), or their incompatibility with particular tokenizers. Motivated by issues raised about the Scikit-learn stop list, we investigate variation among and consistency within 52 popular English-language stop lists, and propose strategies for mitigating these issues.

1 Introduction

Open-source software (OSS) resources tend to become de-facto standards by virtue of their availability and popular use. Resources include tokenization rules and stop word lists, whose precise definitions are essential for reproducible and interpretable models. These resources can be selected somewhat arbitrarily by OSS contributors, such that their popularity within the community may not be a reflection of their quality, universality or suitability for a particular task. Users may then be surprised by behaviors such as the word *computer* being eliminated from their text analysis due to its inclusion in a popular stop list.

This paper brings to the community's attention some issues recently identified in the Scikit-learn stop list. Despite is popular use, the current Scikit-learn maintainers cannot justify the use of this particular list, and are unaware of how it was constructed. This spurs us to investigate variation among and consistency within popular English-language stop lists provided in several popular language processing, retrieval and machine learning libraries. We then make recommendations for improving stop list provision in OSS.

2 Background

Stop words are presumed to be not informative as to the meaning of documents, and hence are defined by being unusually frequent, or by not being "content words". Saif et al. (2014) lists several methods for constructing a stop word list, including: manual construction; words with high *document frequency* or total *term frequency* in a corpus; or by comparing term frequency statistics from a sample of documents with those in a larger collection.[1] In practice, Manning et al. (2008) indicate that statistical approaches tend not to be used alone, but are combined with manual filtering. This paper notes ways in which statistical construction of stop lists may have introduced regrettable errors.

Stop lists have been generated for other languages, such as Chinese (Zou et al., 2006), Thai (Daowadung and Chen, 2012) and Farsi (Sadeghi and Vegas, 2014), using uses similar frequency threshold approaches, are susceptible to the same issues discussed here.

Most prior work focuses on assessing or improving the effectiveness of stop word lists, such as Schofield et al.'s (2017) recent critique of stop lists in topic modeling. Our work instead examines what is available and widely used.

3 Case Study: Scikit-learn

Having become aware of issues with the Scikit-learn (Pedregosa et al., 2011) stop list,[2] we begin by studying it. Scikit-learn provides out-of-the-box feature extraction tools which convert a collection of text documents to a matrix of token counts, optionally removing n-grams containing

[1] They also investigate using supervised feature selection techniques, but the supervised learning context is inapplicable here.

[2] As at version 0.19.1

Proceedings of Workshop for NLP Open Source Software, pages 7–12
Melbourne, Australia, July 20, 2018.

given stop words. Being a popular library for machine learning, many of its users take a naive approach to language processing, and are unlikely to take a nuanced approach to stop word removal.

History While users are able to provide their own stop list, Scikit-learn provides an English-language list since July 2010. The list was initially disabled by default since the contributing author claimed that it did not improve accuracy for text classification (see commit `41b0562`). In November 2010, another contributor argued to enable the list by default, saying that stop word removal is a reasonable default behavior (commit `41128af`). The developers disabled the list by default again in March 2012 (commit `a510d17`).

The list was copied from the Glasgow Information Retrieval Group,[3] but it was unattributed until January 2012 (commit `d4c4c6f`). The list was altered in 2011 to remove the content word *computer* (commit `cdf7df9`), and in 2015 to correct the word *fify* to *fifty* (commit `3e4ebac`).

This history gives a sense of how a stop word list may be selected and provided without great awareness of its content: its provenance was initially disregarded; and some words were eventually deemed inappropriate.

Critique Currently, the list in Scikit-learn has several issues. Firstly, the list is incompatible with the tokenizer provided along with it. It includes words discarded by the default tokenizer, i.e., words less than 2 chars (e.g. *i*), and some abbreviated forms which will be split by the tokenizer (e.g. *hasnt*). What's more, it excludes enclitics generated by the tokenizer (e.g. *ve* of *we've*). In April 2017, a maintainer proposed to add *ve* to the list.[4] Contributors argued this would break reproducibility across software versions, and the issue remains unresolved.

Secondly, there are some controversial words in the list, such as *system* and *cry*. These words are considered to be informative and are seldom included in other stop lists. In March 2018, a user requested the removal of *system* and has gained approval from the community.[5]

Another issue is that the list has some surprising omissions. Compared to extensions of the Glasgow IR list from Stone et al. (2010) used by spaCy (Honnibal and Montani, 2017) and gensim (Řehůřek and Sojka, 2010), the list in Scikit-learn includes modal *has*, but lacks *does*; includes intensifier *very* but excludes *really*; and includes light verb *get* but excludes *make*.

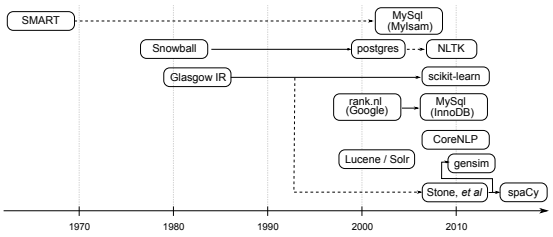

Figure 1: Family tree of popular stop word lists.

The Glasgow IR list appears to have been constructed from corpus statistics, although typographic errors like *fify* suggest manual editing. However, we have not found any documentation about how the Glasgow IR list was constructed. Hence we know little about how to generate comparable lists for other languages or domains.

In the remainder of this paper, we consider how similar issues apply to other open-source stop lists.

4 Datasets

We conduct our experiments on Igor Brigadir's collection of English-language stop word lists.[6] We exclude 1 empty list, 2 lists which contain n-grams ($n > 1$) and 1 list which is intended to augment other lists (i.e. LEMUR's forumstop). Finally, we get 52 lists extracted from various search engines, libraries, and articles. The size of the lists varies (see the right part of Figure 2), from 24 words in the EBSCOhost medical databases list, to 988 words in the ATIRE search engine list.

5 Stop List Families

Through digging into project history, we construct a family tree of some popular stop lists (Figure 1) to show how popular OSS packages adopt or adapt existing lists. Solid lines in the figure correspond to inclusion without major modification, while dashed lines correspond to a more loose adaptation. For instance, the Glasgow IR list used by Scikit-learn was extended with 18 more words by

[3] http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words

[4] https://github.com/scikit-learn/scikit-learn/issues/8687

[5] https://github.com/scikit-learn/scikit-learn/issues/10735

[6] https://github.com/igorbrigadir/stopwords/tree/21fb2ef

Stone et al. (2010), and this list was adopted by OSS packages gensim and spaCy in turn.

A more data driven approach identifies similarities among stop lists by clustering them with the Jaccard distance metric ($JD(A, B) := 1 - \frac{|A \cap B|}{|A \cup B|}$ where A and B are sets of stop words). In Figure 2, we have plotted the same data with a heatmap of word inclusion in order of descending document frequency in the NYT section of Gigaword 5.0 (Parker et al., 2011). Here we take the maximum frequency under three tokenizers from Lucene, Scikit-learn and spaCy. Each of them has different approaches to enclitics (e.g. *hasn't* is treated as *hasn't* in Lucene; *hasn* in Scikit-learn and *has n't* in spaCy).

Looking at the heatmap, we see that stop words are largely concentrated around high document frequency. Some high frequency words are absent from many stop lists because most stop lists assume particular tokenization strategies (See Section 6.2). However, beyond the extremely frequent words, even the shortest lists vary widely in which words they then include. Some stop lists include many relatively low-frequency words. This is most noticeable for large lists like TERRIER and ATIRE-Puurula. TERRIER goes to pains to include synthesized inflectional variants, even *concerninger*, and archaic forms, like *couldst*.

Through the clusermap, we find some lists with very high within-cluster similarity ($JD < 0.2$): Ranks.nl old Google list and MySQL/InnoDB list; PostgreSQL list and NLTK list; Weka list, MALLET list, MySQL-MyISAM list, SMART list and ROUGE list; Glasgow IR list, Scikit-learn list and spaCy/Gensim list. Beyond these simple clusters, some lists appear to have surprisingly high overlap (usually asymmetric): Stanford CoreNLP list appears to be an extension of Snowball's original list; ATIRE-Puurula appears to be an extension of the Ranks.nl Large list.

6 Common Issues for Stop Word Lists

In section 3, we find several issues in the stop word list from Scikit-learn. In this section, we explore how these problems manifest in other lists.

6.1 Controversial Words

We consider words which appear in less than 10% of lists to be controversial.[7] After excluding words which do not begin with English characters, we get 2066 distinct stop words in the 52 lists. Among these words, 1396 (67.6%) words only appear in less than 10% of lists, and 807 (39.1%) words only appear in 1 list (see the bars at the top of Figure 2), indicating that controversial words cover a large proportion. On the contrary, only 64 (3.1%) words are accepted by more than 80% lists. Among the 52 lists, 45 have controversial words.

We further investigate the document frequency of these controversial words using Google Books Ngrams (Michel et al., 2011). Figure 3 shows the document frequency distribution. Note: We scale document frequency of controversial words by the max document frequency among all the words. Although peaked on rare words, some words are frequent (e.g. *general*, *great*, *time*), indicating that the problem is not trivial.

6.2 Tokenization and Stop Lists

Popular software libraries apply different tokenization rules, particularly with respect to word-internal punctuation. By comparing how different stop lists handle the word *doesn't* in Figure 4, we see several approaches: most lists stop *doesn't*. A few stop *doesn* or *doesnt*, but none stop both of these. Two stop *doesn't* as well as *doesnt*, which may help them be robust to different choices of tokenizer, or may be designed to handle informal text where apostrophes may be elided.

However, we find tools providing lists that are inconsistent with their tokenizers. While most lists stop *not*, Penn Treebank-style tokenizers – provided by CoreNLP, spaCy, NLTK and other NLP-oriented packages – also generate the token *n't*. Of our dataset, *n't* is only stopped by CoreNLP.[8] Weka and Scikit-learn both have default tokenizers which delimit tokens at punctuation including ', yet neither stops words like *doesn*.

We find similar results when repeating this analysis on other negated models (e.g. *hasn't*, *haven't*, *wouldn't*), showing that stop lists are often tuned to particular tokenizers, albeit not always the default tokenizer provided by the corresponding package. More generally, we have not found any OSS package which documents how tokenization relates to the choice of stop list.

[7] Some false negatives will result from the shared origins of lists detailed in the previous section, but we find very similar results if we remove near-duplicate lists (Jaccard distance < 0.2) from our experiments.

[8] We are aware that spaCy, in commit `f708d74`, recently amended its list to improve consistency with its tokenizer, adding *n't* among other Penn Treebank contraction tokens.

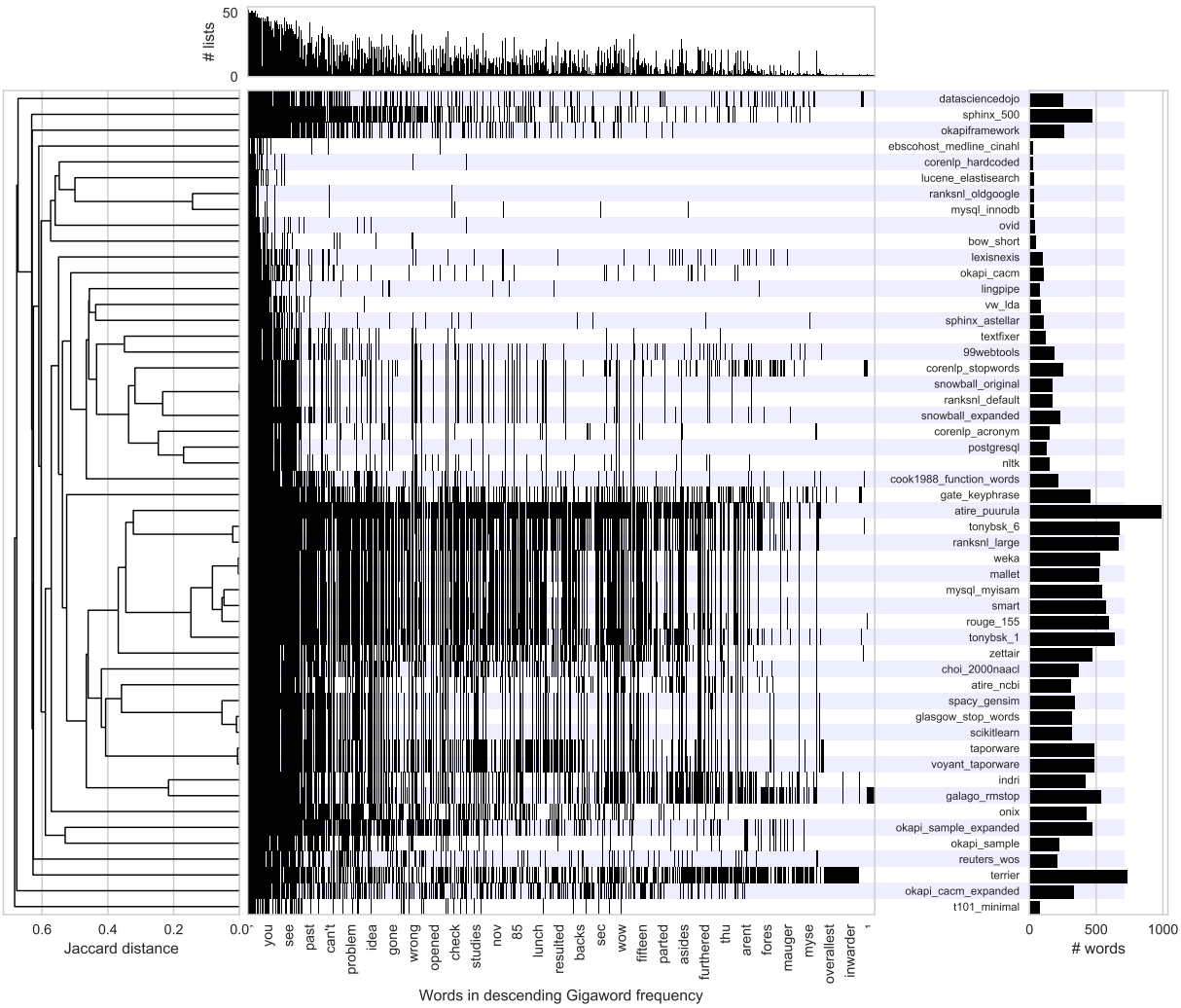

Figure 2: Word inclusion in clustered English stop word lists. Words are ordered by descending document frequency. The dendrogram on the left indicates minimum Jaccard distance between stop list pairs when merged. The bars on the right show the number of words in each list, and the bars on the top indicate the number of lists each word is found in.

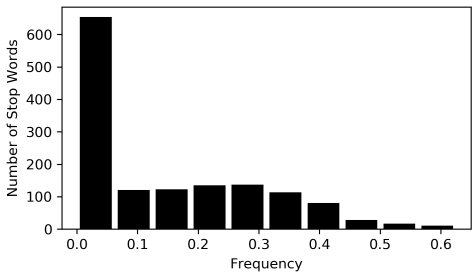

Figure 3: Document frequency distribution of controversial words

6.3 Incompleteness

Stop lists generated exclusively from corpus statistics are bound to omit some inflectional forms of an included word, as well as related lexemes, such as less frequent members of a functional syntactic class. In particular, stop word list construction prefers frequency criteria over contextual indicators of a word's function, despite Harris's (1954) well-established theory that similar words (e.g. function words, light verbs, negated modals) should appear in similar contexts.

To continue the example of negated modals, we find inconsistencies in the inclusion of *have* and its variants, summarized in Figure 5. Weka includes *has*, but omits its negated forms, despite including *not*. Conversely, Okapi includes *doesnt*, but omits

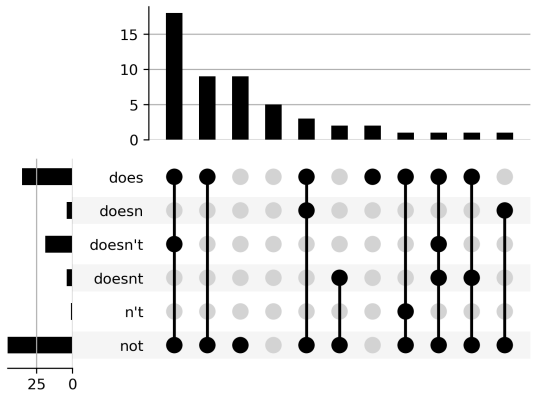

Figure 4: Number of stop lists that include variants of *doesn't* and their combinations.

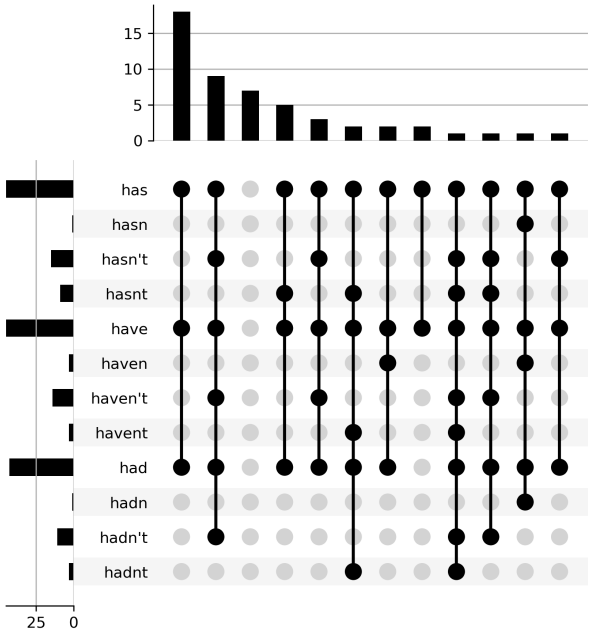

Figure 5: Number of stop lists that include variants of *have* and their combinations.

does. Several lists include *has*, *hasnt*, *have* and *had*, but omits *havent* and *hadnt*. Several lists that include *has* and *have* forms omit *had* forms. These inclusions and omissions seem arbitrary.

Some negated modals like *shan't* and *mustn't* are absent more often than other modals (e.g. *doesn't*, *hasn't*), which may be an unsurprising artifact of their frequency, or may be an ostensive omission because they are more marked.

TERRIER list (Ounis et al., 2005) appears to have generated inflectional variants, to the extent of including *concerninger*. This generally seems an advisable path towards improved consistency.

7 Improving Stop List Provision in OSS

Based on the analysis above, we propose strategies for better provision of stop lists in OSS:

Documentation Stop lists should be documented with their assumptions about tokenization and other limitations (e.g. genre). Documentation should also include information on provenance and how the list was built.

Dynamic Adaptation Stop lists can be adapted dynamically to match the NLP pipeline. For example, stop lists can be adjusted according to the tokenizer chosen by the user (e.g. through applying the tokenizer to the stop list); a word which is an inflectional variant of a stop word could also be removed

Quality Control The community should develop tools for identifying controversial terms in stop lists (e.g. words that are frequent in one corpus but infrequent in another), and to assist in assessing or mitigating incompleteness issues. For instance, future work could evaluate whether the nearest neighborhood of stop words in vector space can be used to identify incompleteness.

Tools for Automatic Generation A major limitation of published stop lists is their inapplicability to new domains and languages. We thus advocate language independent tools to assist in generating new lists, which could incorporate the quality control tools above.

8 Conclusion

Stop word lists are a simple but useful tool for managing noise, with ubiquitous support in natural language processing software. We have found that popular stop lists, which users often apply blindly, may suffer from surprising omissions and inclusions, or their incompatibility with particular tokenizers. Many of these issues may derive from generating stop lists using corpus statistics. We hence recommend better documentation, dynamically adapting stop lists during preprocessing, as well as creating tools for stop list quality control and automatically generating stop lists.

Acknowledgments

We thank Igor Brigadir for collecting and providing English-language stop word lists along with their provenance. We also thank Scikit-learn contributors for bringing these issues to our attention.

References

P. Daowadung and Y. H. Chen. 2012. Stop word in readability assessment of thai text. In *Proceedings of 2012 IEEE 12th International Conference on Advanced Learning Technologies*, pages 497–499.

Zelig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. 2005. Terrier information retrieval platform. In *Proceedings of the 27th European Conference on Advances in Information Retrieval Research*, pages 517–519.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword fifth edition LDC2011T07. Linguistic Data Consortium.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Mohammad Sadeghi and Jess Vegas. 2014. Automatic identification of light stop words for persian information retrieval systems. *Journal of Information Science*, 40(4):476–487.

Hassan Saif, Miriam Fernández, Yulan He, and Harith Alani. 2014. On stopwords, filtering and data sparsity for sentiment analysis of Twitter. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation. Proceedings.*, pages 810–817.

Alexandra Schofield, Måns Magnusson, and David Mimno. 2017. Pulling out the stops: Rethinking stopword removal for topic models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 432–436.

Benjamin Stone, Simon Dennis, and Peter J. Kwantes. 2010. Comparing methods for single paragraph similarity analysis. *Topics in Cognitive Science*, 3(1):92–122.

Feng Zou, Fu Lee Wang, Xiaotie Deng, Song Han, and Lu Sheng Wang. 2006. Automatic construction of chinese stop word list. In *Proceedings of the 5th WSEAS International Conference on Applied Computer Science*.