The Journal of Open Source Software

niacin: A Python package for text data enrichment

Dillon Niederhut[1]

1 Novi Labs

DOI: 10.21105/joss.02136

Software

- Review
- Repository
- Archive

Editor: Olivia Guest

Reviewers:

- @BayesforDays
- @sara-02

Submitted: 19 February 2020
Published: 06 June 2020

Summary

A common component of frameworks for building robust and accurate learning models is a utility for performing sets of perturbations on the input data. In machine vision, for example, rotating, cropping, stretching, warping, and flipping training images have all been shown to increase model accuracy, and reduce certain kinds of overfitting (Chen, Dobriban, & Lee, 2019). Popular machine vision libraries like PyTorch, Tensorflow, and FastAI all have built-in utilities for performing these transformations (Abadi et al., 2015; Howard & Gugger, 2020; Paszke et al., 2019).

In the domain of NLP, there is an analogous set of transformations that have been shown to increase model accuracy. Common transformations include adding superfluous whitespace, swapping an individual word for a synonym, replacing a word with a misspelled version, and round-tripping a sentence through a translation model into another language and back again (Coulombe, 2018; Zhang, Zhao, & LeCun, 2015). Some transforms, like misspellings or synonyms, attempt to directly model an unobserved but common change in the input features. Others, like adding a space in the middle of a word, act similarly to dropout for word-level models, in that they effectively remove the word as an input (e.g. the two halves of the word are unlikely to be present in the vocabulary).

For specific tasks in NLP, principally in language modeling, model improvement can be effected with incorrect examples of input data, so that the model learns not to produce them. This practice is called negative sampling, and was used in the development of both word2vec (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) and GLoVE (Pennington, Socher, & Manning, 2014). This can be done with words sampled from the whole vocabulary, or in a more targeted fashion by including related, but inappropriate tokens. For example, in English, "I ate pizza for lunch" is a valid sentence. Replacing "pizza" and "lunch" with a higher level category (these are called hypernyms), produces – "I ate *dish* for *meal*" an output that is technically well-formed, but violates the Gricean expectation of utility.

There are also sets of transforms whose aim is to make models more robust in the face of adversarial inputs. These transforms include replacing letters with symbols that are visually similar, adding nuisance characters around words, adding random characters at the end of the document, and appending the word "love" to the end of a sentence (Gröndahl, Pajola, Juuti, Conti, & Asokan, 2018). Adding nuisance characters around words, or replacing characters with symbols, disrupt the behavior of vocabulary-based models, in that they will not know how to encode an input like "sch00l", although it is visually clear to an experienced reader of English that this is the word "school". Adding words with very positive or very negative connotations have been shown to be an effective way to disrupt sentiment classifiers, which often rely on counting the number of times such words appear.

Here, we introduce niacin, a python library for performing a large set of these commonly-used text transformations on a given input, with some probability. For example, a commonly used text augmentation technique is to replace individual words in a sentence with ones that are closely related. The idea here is that the sentences "I am unhappy" and "I am sad" convey a similar semantic value, even though the individual words are different. In niacin, you can use a function called add_synonyms for this, which, at a replacement probability of 1.0:

```
>>> add_synonyms("this is the song that never ends and it goes on
and on my friends", p=1.0)
```

produces outputs like this:

> this is the *vocal* that *ne'er terminate* and it *go away* on and on my friend

There are also functions for replacing words with common misspellings:

```
>>> add_misspelling("There are also functions for replacing words with
their common misspellings", 1.0)
```

> There are *aslo* functions for replacing words *withh thier* common misspellings

replacing contractions:

```
>>> add_contractions('You are rad', 1.0)
```

> *you're* rad

and substituting numbers and symbols for letters they resemble:

```
>>> add_leet("Hello, how are you?", 1.0)
```

> *H3110*, *h0w r u*?

When possible, function documentation links back to the paper which originally suggested or implemented the transformation, should a user want more information about how apply the function in practice. For example, replacing every single character in your text input will render it useless, but there is emprical work suggesting that no more than a 10% probability of replacement should be used. Data sources (principally WordNet (Miller, 1995), but also Wikipedia) are always cited.

In 2018, when niacin was first published online, there were a small number of libraries that contain text augmentation functions, for example noisemix and EDA (Wei & Zou, 2019). In general, these were published only as a part of a conference paper, and were not actively maintained. EDA, for example, contains the warning "The code is not documented". They also tended to be limited to the scope of the transformations included in the initial paper. There is not an exact one-to-one correspondence between all the capabilities of these libraries, but this table shows an approximation of comparing features across them:

transformation	niacin	noisemix	eda
synonym replacement	x	x	x
hypernym replacement	x		
hyponym replacement	x		
swap word order	x	x	x
swap character order	x	x	
remove articles	x		

transformation	niacin	noisemix	eda
contract/expand word	x		
remove punctuation	x	x	
add/delete space	x	x	
add/delete letter	x	x	
add typo	x	x	
add misspelling	x		
wrap in parens	x		
add applause emoji	x		
add 'love'	x		
add random characters	x		
add backtranslation	x		

More recently, other software tools have been published that cover some parts of niacin's functionality, and extend further in other areas. `nlpaug`, for example, includes large language models like BERT that can perform synonym replacement in a way that is context-dependent – something that wordnet cannot do (Ma, 2019). However, it does not include some of the character-based transformations, nor the adversarial ones. `TextAttack` implements several recently published methods in model-based adversarial attacks on language models, in both the white-box and black-box domain, and includes a small number of augmentation techniques in addition (Morris, Lifland, Yoo, & Qi, 2020).

It is our hope that having a collection of already-implemented transformations with a uniform interface will make it easy for researchers to include them in their own data processing pipelines.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/

Chen, S., Dobriban, E., & Lee, J. H. (2019). Invariance reduces variance: Understanding data augmentation in deep learning and beyond. Retrieved from http://arxiv.org/abs/1907.10905

Coulombe, C. (2018). Text data augmentation made simple by leveraging nlp cloud apis. Retrieved from http://arxiv.org/abs/1812.04718

Gröndahl, T., Pajola, L., Juuti, M., Conti, M., & Asokan, N. (2018). All you need is "love": Evading hate-speech detection. Retrieved from http://arxiv.org/abs/1808.09115

Howard, J., & Gugger, S. (2020). Fastai: A layered api for deep learning. *Information*, *11*(2), 108. doi:10.3390/info11020108

Ma, E. (2019). NLP augmentation. https://github.com/makcedward/nlpaug.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Retrieved from http://arxiv.org/abs/1310.4546

Miller, G. (1995). WordNet: A lexical database for english. *Communications of the ACM*, *38*, 39–41.

Morris, J. X., Lifland, E., Yoo, J. Y., & Qi, Y. (2020). TextAttack: A framework for adversarial attacks in natural language processing.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics. doi:10.3115/v1/D14-1162

Wei, J., & Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 6382–6388). Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1670

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. Retrieved from http://arxiv.org/abs/1509.01626