wildnlp Documentation

Release 1.0.1 - Beta

Dominika

Apr 01, 2019

Contents

Performance of NLP models should be measured not only in terms of well established metrics if we want to use them in the real life.

This module contains implementation of multiple functions designed to corrupt text in a way resembling naturally occuring mistakes (aspects).

CHAPTER 1

Quick Start Guide

5 minutes is enough to start using the module!

See how easy you can enrich your analysis of a NLP model with a robustness test.

1.1 Installation

```
pip install wild-nlp
```

1.2 Loading a dataset

```
from wildnlp.datasets import SampleDataset

dataset = SampleDataset()
dataset.load()
```

1.3 Corrupting a text

There are two usecases. You may either apply corruption to a supported dataset or modify an arbtrary string.

1.3.1 Applying corruption to a supported dataset

```
from wildnlp.aspects.dummy import Reverser

modified = dataset.apply(Reverser())
```

1.3.2 Modifying a string

```
from wildnlp.aspects.dummy import Reverser

modified = Reverser()('A string to be modified.')
```

Note: All instances of classes derived from the Aspect class are callable. You can think of them as any other functions.

```
from wildnlp.aspects.dummy import Reverser

reverser_object = Reverser()
modified = reverser_object('A string to be modified.')

# Note that this is the same as
# modified = Reverser()('A string to be modified.')
```

1.4 Full example with multiple corruptors

1.4.1 The code

```
from wildnlp.aspects.dummy import Reverser, PigLatin
from wildnlp.aspects.utils import compose
from wildnlp.datasets import SampleDataset

# Create a dataset object and load the dataset
dataset = SampleDataset()
dataset.load()

# Crate a composed corruptor function.
# Functions will be applied in the same order they appear.
composed = compose(Reverser(), PigLatin())

# Apply the function to the dataset
modified = dataset.apply(composed)
```

1.4.2 The dataset's contents

```
["Manning is a leader in applying Deep Learning "
 "to Natural Language Processing",
 "Manning has coauthored leading textbooks on statistical "
 "approaches to Natural Language Processing"]
```

1.4.3 After applying the aspects

```
>>> print(modified)
['ninnamgay isay aay edaelray inay niylppagay eedpay ninraelgay toay arutanlay␣
↪gaugnaleay nissecorpgay', 'ninnamgay ahsay erohtuaocday nidaelgay koobtxetsay onay␣
↪acitsitatslay ehcaorppasay toay arutanlay gaugnaleay nissecorpgay']
```

1.5 Saving the dataset

Serialized dataset will have exactly the same format as the original dataset before modification.

It means that you don't have to modify your existing code to test robustness of your models. Simply modify a dataset, save the modified version and provide it as an input to your existing pipeline instead of the original file.

Note: in this example no file will be saved.

```
from wildnlp.datasets import SampleDataset

dataset = SampleDataset()
dataset.load()

dataset.save(data.data, '<path_to_file>')
```

CHAPTER 2

Aspects

Functions that can be applied to sentences to corrupt them in a controled way. Corrupted sentences can be then used to test NLP models' robustness.

2.1 Base class

class wildnlp.aspects.base.**Aspect**

Base, abstract class. All the aspects must implement the __call__ method.

__call__(*sentence*)

We want to directly call objects of the Aspect class for easy chaining. This function will be applied to sentences.

static _detokenize(*tokens*)

Join tokens into tokens including punctuation and special characters.

Parameters **tokens** – List of tokens.

Returns A sentence as a single string.

static _tokenize(*sentence*)

Split text into tokens including punctuation and special characters.

Parameters **sentence** – A sentences as a single string.

Returns List of tokens.

2.2 Utility functions

wildnlp.aspects.utils.**compose**(**functions*)

Chains multiple aspects into a single function.

Parameters **functions** – Object(s) of the Callable instance.

Returns chained function

Example:

```
from wildnlp.aspects.utils import compose
from wildnlp.aspects import Swap, QWERTY

composed_aspect = compose(Swap(), QWERTY())
modified_text = composed_aspect('Text to corrupt')
```

2.3 Articles

class wildnlp.aspects.articles.**Articles**(*swap_probability=0.5*, *seed=42*)

Bases: *wildnlp.aspects.base.Aspect*

Randomly removes or swaps articles into wrong ones.

Caution: Uses random numbers, default seed is 42.

__init__(*swap_probability=0.5*, *seed=42*)

Parameters

- **swap_probability** – Probability of applying a transformation. Defaults to 0.5.
- **seed** – Random seed.

2.4 Characters removal

class wildnlp.aspects.remove_char.**RemoveChar**(*char=None*, *words_percentage=50*, *characters_percentage=10*, *seed=42*)

Bases: *wildnlp.aspects.base.Aspect*

Randomly removes characters from words.

Note: Note that you may specify white space as a character to be removed but it'll be processed differently.

Caution: Uses random numbers, default seed is 42.

__init__(*char=None*, *words_percentage=50*, *characters_percentage=10*, *seed=42*)

Parameters

- **words_percentage** – Percentage of words in a sentence that should be transformed. If greater than 0, always at least single word will be transformed.
- **characters_percentage** – Percentage of characters in a word that should be transformed. If greater than 0 always at least single character will be transformed.
- **char** – If specified only that character will be randomly removed. The specified character can also be a white space.

- **seed** – Random seed.

2.5 Characters swapping

class wildnlp.aspects.swap.**Swap**(*transform_percentage=100*, *seed=42*)
Bases: *wildnlp.aspects.base.Aspect*

Randomly swaps two characters within a word, excluding punctuations. It's possible that the same two characters will be swapped, so the word won't be changed, for example *letter* can become *letter* after swapping.

> **Caution:** Uses random numbers, default seed is 42.

__init__(*transform_percentage=100*, *seed=42*)

Parameters

- **transform_percentage** – Maximum percentage of words in a sentence that should be transformed.
- **seed** – Random seed.

2.6 Digits2Words

class wildnlp.aspects.digits2words.**Digits2Words**
Bases: *wildnlp.aspects.base.Aspect*

Converts numbers into words. Handles floating numbers as well.

All numbers will be converted

2.7 Misspelling

class wildnlp.aspects.misspelling.**Misspelling**(*use_homophones=False*, *seed=42*)
Bases: *wildnlp.aspects.base.Aspect*

Misspells words appearing in the Wikipedia list of **commonly misspelled English words** (default): https://en.wikipedia.org/wiki/Commonly_misspelled_English_words

> **Tip:** You can use **homophones** instead: https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/Homophones

If a word has more then one common misspelling, the replacement is selected randomly.

All words that have any misspellings listed will be replaced.

> **Caution:** Uses random numbers, default seed is 42.

__init__(*use_homophones=False*, *seed=42*)

Parameters

- **use_homophones** – If True list of homophones will be used to replace words.
- **seed** – Random seed.

2.8 Punctuation

class wildnlp.aspects.punctuation.**Punctuation**(*char=', ', add_percentage=0, remove_percentage=100, seed=42*)

Bases: *wildnlp.aspects.base.Aspect*

Randomly adds or removes specified punctuation marks. The implementation guarantees that punctuation marks won't be appended to the original ones or won't replace them after removal.

With default settings all occurrences of the specified punctuation mark will be removed.

- Example:

```
Sentence, have a comma.

Possible transformations:
- Sentence have, a comma.
- Sentence, have, a, comma.

Impossible transformations:
- Sentence,, have a comma.
```

Caution: Uses random numbers, default seed is 42.

__init__(*char=', ', add_percentage=0, remove_percentage=100, seed=42*)

Parameters

- **char** – Punctuation mark that will be removed or added to sentences.
- **add_percentage** – Max percentage of white spaces in a sentence to be prepended with punctuation marks.
- **remove_percentage** – Max percentage of existing punctuation marks that will be removed.
- **seed** – Random seed.

2.9 QWERTY

class wildnlp.aspects.qwerty.**QWERTY**(*words_percentage=1, characters_percentage=10, seed=42*)

Bases: *wildnlp.aspects.base.Aspect*

Simulates errors made while writing on a QWERTY-type keyboard. Characters are swapped with their neighbors on the keyboard.

Caution: Uses random numbers, default seed is 42.

__init__(*words_percentage=1, characters_percentage=10, seed=42*)

Parameters

- **words_percentage** – Percentage of words in a sentence that should be transformed. If greater than 0, always at least single word will be transformed.
- **characters_percentage** – Percentage of characters in a word that should be transformed. If greater than 0 always at least single character will be transformed.
- **seed** – Random seed.

2.10 Sentiment words masking

class wildnlp.aspects.sentiment_masking.**SentimentMasking**(*char='*'*, *use_positive=False*, *seed=42*)

Bases: *wildnlp.aspects.base.Aspect*

This aspect reflects attempts made by Internet users to mask profanity or hate speech in online forums to evade moderation. We perform masking (replacing random, single character with for example an asterisk) of negative (or positive for completeness) words from Opinion Lexicon: http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

All words that are listed will be transformed.

> **Caution:** Uses random numbers, default seed is 42.

__init__(*char='*'*, *use_positive=False*, *seed=42*)

Parameters

- **char** – A character that will be used to mask words.
- **use_positive** – If True positive (instead of negative) words will be masked.
- **seed** – Random seed.

CHAPTER 3

Datasets

Details regarding popular datasets for various NLP problems that are supported by the wild-nlp.

3.1 Base class

class wildnlp.datasets.base.**Dataset**(**args*, ***kwargs*)

apply(**args*, ***kwargs*)
The method should iterate through texts in the dataset and apply a given aspect to them.

data
Property of the Dataset class.

Returns Internal object storing a loaded dataset.

load(**args*, ***kwargs*)
The method should handle loading and parsing of a specific dataset.

save(**args*, ***kwargs*)
The method should iterate through texts in the dataset and apply a given aspect to them.

3.2 CoNLL 2003

class wildnlp.datasets.conll.**CoNLL**(**args*, ***kwargs*)
Bases: *wildnlp.datasets.base.Dataset*

The CoNLL-2003 shared task data for language-independent named entity recognition. For details see: https://www.clips.uantwerpen.be/conll2003/ner/

apply(*aspect*, *apply_to_ne=False*)

Parameters

- **aspect** – transformation function
- **apply_to_ne** – if *False*, transformation won't be applied to Named Entities. If *True*, transformation will be applied only to Named Entities.

Returns modified dataset in the following form:

```
[{tokens: array(<tokens>)
  pos_tags: array(<pos_tags>),
  chunk_tags: array(<chunk_tags>),
  ner_tags: array(<ner_tags>},

  ...,

  ]
```

load(*path*)

Reads a CoNLL dataset file and loads into internal data structure in the following form:

```
[{tokens: array(<tokens>)
  pos_tags: array(<pos_tags>),
  chunk_tags: array(<chunk_tags>),
  ner_tags: array(<ner_tags>},

  ...,

  ]
```

Parameters **path** – A path to a file with CoNLL data

Returns None

save(*data*, *path*)

Saves data in the CoNLL format

Parameters **data** – list of dictionaries in the following form:

```
[{tokens: array(<tokens>)
  pos_tags: array(<pos_tags>),
  chunk_tags: array(<chunk_tags>),
  ner_tags: array(<ner_tags>},

  ...,

  ]
```

Parameters **path** – Path to save the file. If the file exists, it will be overwritten.

Returns None

3.3 SNLI

class wildnlp.datasets.snli.**SNLI**(**args*, ***kwargs*)

Bases: *wildnlp.datasets.base.Dataset*

The SNLI dataset supporting the task of natural language inference. For details see: https://nlp.stanford.edu/projects/snli/

apply(*aspect*)
Modifies premises (sentence1) in the dataset leaving other data intact.

load(*path*)
Loads a SNLI dataset.

Parameters **path** – A path to a SNLI data file in JSONL format.

Returns None

save(*data*, *path*)
Saves data in the SNLI format

3.4 SQuAD

class wildnlp.datasets.squad.**SQuAD**
Bases: *wildnlp.datasets.base.Dataset*

The SQuAD dataset. For details see: https://rajpurkar.github.io/SQuAD-explorer/

apply(*aspect*)
Modifies questions in the dataset leaving other data intact.

load(*path*)
Loads a SQuAD dataset.

Parameters **path** – A path to a SQuAD data file in JSONL format.

Returns None

save(*data*, *path*)
Saves data in the SQuAD format

3.5 IMDB

class wildnlp.datasets.imdb.**IMDB**(**args*, ***kwargs*)
Bases: *wildnlp.datasets.base.Dataset*

The IMDB dataset containing movie reviews for a sentiment analysis. The dataset consists of 50 000 reviews of two classes, negative and positive. Each review is stored in a separate text file. For details see: http://ai.stanford.edu/~amaas/data/sentiment/

apply(*aspect*)
Modifies contents of the whole files in the IMDB dataset.

load(*path*)
Loads a SNLI dataset.

Parameters **path** – A path to single file, directory containing review files or list of paths to such directories.

Returns None

save(*data*, *path*)
Saves IMDB reviews to separate files with the original names.

Parameters **path** – path to a top directory where files will be saved.

Returns None

save_tsv(*data*, *path*)
Convenience function for saving IMDB reviews into a single TSV file.

Parameters **path** – Path to a tab separated file.

Returns None

CHAPTER 4

How to contribute

Already implemented functions by no means exhaust all the possibilities for corruptin a text.

There are also many popular datasets that we still don't support.

If you'd like to extend the module, you're more than welcome! :)

4.1 General remarks

1. Unit tests are boring but sometimes also helpful.
2. Documenting your work helps other to use it.

4.2 How to add an Aspect

All the aspects inherits from the Aspect class. The only thin you should remeber is to implement the __call__ function accepting a single argument (string) and returning a single output (string).

That's it, we leave the details to you! Remeber that natural language is tricky, there are punctuation marks, capitalizations, apostrophes etc, that you may would like to leave intact if it's not the target of your aspect.

Caution: Every aspect must implement __call__ function accepting a string as an input and outputing a string.

4.3 How to add a dataset support

All the datasets inherits from the Dataset class. There are only 3 methods that it must implement.

1. **load** - load a dataset from file to an iterable internal object.

2. **apply** - iterate through all elements (strings) in a dataset and modify them one by one.

3. **save** - save the dataset in the same exact format as the original one.

Caution: Saved dataset should have the same format as the original one.

Python Module Index

w

Index

Symbols

A

C

D

I

L

M

P

Q

R

S

W