LanguageFlow Documentation

Release 1.1.6

Vu Anh

Dec 20, 2017

Notes

1 LanguageFlow 3

2 Installation 5

3 Usage 7

4 Contributing 9

5 Credits 13

6 History 15

7 languageflow 17

8 languageflow.transformer 19

9 languageflow.model 21

10 languageflow.log 25

11 Indices and tables 27

Python Module Index 29

Data loaders and abstractions for text and NLP

CHAPTER 1

LanguageFlow

Data loaders and abstractions for text and NLP

- Free software: GNU General Public License v3
- Documentation: link

1.1 Prerequisite

Install dependencies

```
$ pip install fasttext
```

1.2 Installation

Stable version

```
$ pip install https://github.com/undertheseanlp/languageflow/archive/master.zip
```

Latest version

```
$ pip install https://github.com/undertheseanlp/languageflow/archive/develop.zip
```

CHAPTER 2

Installation

2.1 Stable release

To install LanguageFlow, run this command in your terminal:

```
$ pip install languageflow
```

This is the preferred method to install LanguageFlow, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for LanguageFlow can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/undertheseanlp/languageflow
```

Or download the tarball:

```
$ curl  -OL https://github.com/undertheseanlp/languageflow/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use LanguageFlow in a project:

```
import languageflow
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/undertheseanlp/languageflow/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

LanguageFlow could always use more documentation, whether as part of the official LanguageFlow docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/underthesea/lanugageflow/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *languageflow* for local development.

1. Fork the *languageflow* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/languageflow.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv languageflow
$ cd languageflow/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 languageflow tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/undertheseanlp/languageflow/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_languageflow
```

CHAPTER 5

Credits

5.1 Development Lead

- Vu Anh <brother.rain.1024@gmail.com>

5.2 Contributors

- Bui Nhat Anh <buinhatanh1208@gmail.com>

CHAPTER 6

History

6.1 0.1.0 (2017-09-18)

- First release on PyPI.

CHAPTER 7

languageflow

7.1 Flow

class languageflow.flow.**Flow**
Pipeline to build a model

Examples

```
>>> from languageflow.flow import Flow
>>> flow = Flow()
>>> flow.data(X, y)
>>> flow.transform(TfidfTransformer())
>>> model = Model(SGD(), "SGD")
>>> flow.add_model(model)
>>> flow.train()
```

add_model(*model*)
Add model to flow

add_score(*score*)

data(*X=None*, *y=None*, *sentences=None*)
Add data to flow

export(*model_name*, *export_folder*)
Export model and transformers to export_folder

Parameters

- **model_name** (*string*) – name of model to export
- **export_folder** (*string*) – folder to store exported model and transformers

set_learning_curve(*start*, *stop*, *offset*)

set_validation(*validation*)

train()
Train model with transformed data

transform(*transformer*)
Add transformer to flow and apply transformer to data in flow

Parameters **transformer** (*Transformer*) – a transformer to transform data

CHAPTER 8

languageflow.transformer

8.1 NumberRemover

class languageflow.transformer.number.**NumberRemover**

Remove numbers in documents

transform(*raw_documents*)

Remove number in each document

Parameters **raw_documents** (*iterable*) – An iterable which yields either str, unicode

Returns **X** – cleaned documents

Return type iterable

CHAPTER 9

languageflow.model

9.1 SGDClassifier

languageflow.model.sgd.**SGDClassifier**

9.2 XGBoostClassifier

languageflow.model.xgboost.**XGBoostClassifier**

9.3 KimCNNClassifier

class languageflow.model.cnn.**KimCNNClassifier**(*batch_size=50, kernel_sizes=[3, 4, 5], num_kernel=100, embedding_dim=50, epoch=50, lr=0.001*)

An implementation of the model from Kim2014 paper

Parameters

- **batch_size** (*int*) – Number of samples per gradient update
- **kernel_sizes** (*list of int*) –
- **num_kernel** (*int*) –
- **embedding_dim** (*int*) – only for CNN-rand
- **epoch** (*int*) – Number of epochs to train the model
- **lr** (*float, optional*) – Learning rate (default: 1e-3)

Examples

```
>>> from languageflow.flow import Flow
>>> flow = Flow()
>>> flow.data(X, y)
>>> model = Model(KimCNNClassifier(batch_size=5, epoch=150, embedding_dim=300)
>>> flow.add_model(model, "KimCNNClassifier"))
>>> flow.train()
```

fit (*X*, *y*)
Fit KimCNNClassifier according to X, y

Parameters

- **X** (*list of string*) – each item is a raw text
- **y** (*list of string*) – each item is a label

predict (*X*)

Parameters **X** (*list of string*) – Raw texts

Returns **C** – List labels

Return type list of string

9.4 FastTextClassifier

class languageflow.model.fasttext.**FastTextClassifier**
Only support multiclass classification

fit (*X*, *y*, *model_filename=None*)
Fit FastText according to X, y

Parameters

- **X** (*list of string*) – each item is a raw text
- **y** (*list of string*) – each item is a label

predict (*X*)
In order to obtain the most likely label for a list of text

Parameters **X** (*list of string*) – Raw texts

Returns **C** – List labels

Return type list of string

9.5 CRF

class languageflow.model.crf.**CRF** (*params=None*)

fit (*X*, *y*)
Fit CRF according to X, y

Parameters

- **X** (*list of text*) – each item is a text

- **y** (*list*) – each item is either a label (in multi class problem) or list of labels (in multi label problem)

predict(*X*)

Predict class labels for samples in X.

Parameters **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Samples.

CHAPTER 10

languageflow.log

Analyze and save test results.

10.1 MulticlassLogger

class languageflow.log.multiclass.**MulticlassLogger**

Analyze and save multiclass results

static log(*X_test*, *y_test*, *y_pred*, *folder*)

Parameters

- **X_test** (*list of string*) – Raw texts
- **y_test** (*list of string*) – Test labels
- **y_pred** (*list of string*) – Predict labels
- **folder** (*string*) – log folder

10.2 MultilabelLogger

class languageflow.log.multilabel.**MultilabelLogger**

Analyze and save multilabel results to multilabel.json and result.json files

static log(*X_test*, *y_test*, *y_pred*, *log_folder*)

Parameters

- **X_test** (*list of string*) – Raw texts
- **y_test** (*list of string*) – Test labels
- **y_pred** (*list of string*) – Predict labels
- **log_folder** (*string*) – path to log folder

10.3 TfidfLogger

class languageflow.log.tfidf.**TfidfLogger**
Analyze and save tfidf results

static log(*model_folder*, *binary_file='tfidf.transformer.bin'*, *log_folder='analyze'*)

Parameters

- **model_folder** (*string*) – folder contains binaries file of model
- **binary_file** (*string*) – file path to tfidf binary file
- **log_folder** (*string*) – log folder

10.4 CountLogger

class languageflow.log.count.**CountLogger**
Analyze and save tfidf results

static log(*model_folder*, *binary_file='count.transformer.bin'*, *log_folder='analyze'*)

Parameters

- **model_folder** (*string*) – folder contains binaries file of model
- **binary_file** (*string*) – file path to count transformer binary file
- **log_folder** (*string*) – log folder

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

languageflow.log, 25
languageflow.model, 21
languageflow.transformer, 19

Index

A

add_model() (languageflow.flow.Flow method), 17
add_score() (languageflow.flow.Flow method), 17

C

CountLogger (class in languageflow.log.count), 26
CRF (class in languageflow.model.crf), 22

D

data() (languageflow.flow.Flow method), 17

E

export() (languageflow.flow.Flow method), 17

F

FastTextClassifier (class in languageflow.model.fasttext), 22
fit() (languageflow.model.cnn.KimCNNClassifier method), 22
fit() (languageflow.model.crf.CRF method), 22
fit() (languageflow.model.fasttext.FastTextClassifier method), 22
Flow (class in languageflow.flow), 17

K

KimCNNClassifier (class in languageflow.model.cnn), 21

L

languageflow.log (module), 25
languageflow.model (module), 21
languageflow.transformer (module), 19
log() (languageflow.log.count.CountLogger static method), 26
log() (languageflow.log.multiclass.MulticlassLogger static method), 25
log() (languageflow.log.multilabel.MultilabelLogger static method), 25
log() (languageflow.log.tfidf.TfidfLogger static method), 26

M

MulticlassLogger (class in languageflow.log.multiclass), 25
MultilabelLogger (class in languageflow.log.multilabel), 25

N

NumberRemover (class in languageflow.transformer.number), 19

P

predict() (languageflow.model.cnn.KimCNNClassifier method), 22
predict() (languageflow.model.crf.CRF method), 23
predict() (languageflow.model.fasttext.FastTextClassifier method), 22

S

set_learning_curve() (languageflow.flow.Flow method), 17
set_validation() (languageflow.flow.Flow method), 17
SGDClassifier (in module languageflow.model.sgd), 21

T

TfidfLogger (class in languageflow.log.tfidf), 26
train() (languageflow.flow.Flow method), 18
transform() (languageflow.flow.Flow method), 18
transform() (languageflow.transformer.number.NumberRemover method), 19

X

XGBoostClassifier (in module languageflow.model.xgboost), 21