用于Facebook fastText的Python接口

用于Facebook fastText的Python接口

推荐 0 推荐

收藏 0 收藏，2986 浏览

详细内容 评论 57 同类相比 1203 发布的版本 v0.7.6

学习教程

社区推荐：掘金是一个面向开发者的技术社区，内容涵盖了前后端，Android 和 iOS 开发，每天更新深度文章，最新技术资讯，优质开源库推送。无论是入门还是进阶，掘金都可以帮助你成为更优秀的开发者。

fasttext

build passing pypi package 0.8.3

fasttext is a Python interface for Facebook fastText.

Requirements

fasttext support Python 2.6 or newer. It requires Cython in order to build the C++ extension.

Installation

```
pip install fasttext
```

Example usage

This package has two main use cases: word representation learning and text classification.

These were described in the two papers 1 and 2.

热门度与活跃度

2.3 0.0

最新发布的版本

版本：**v0.7.6**

Source code(tar.gz)

Source code(zip)

发布时间:**1年前**

访问GitHub主页

Watchers：**48**

Star：**772**

Fork：**176**

创建时间: **2016-08-07 02:09:11**

最后Commits： **6月前**

许可协议：**BSD-3-Clause**

Word representation learning

In order to learn word vectors, as described in 1, we can use `fasttext.skipgram` and `fasttext.cbow` function like the following:

```
import fasttext

# Skipgram model
model = fasttext.skipgram('data.txt', 'model')
print model.words # list of words in dictionary

# CBOW model
model = fasttext.cbow('data.txt', 'model')
print model.words # list of words in dictionary
```

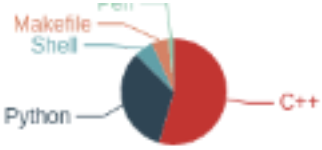
where `data.txt` is a training file containing `utf-8` encoded text. By default the word vectors will take into account character n-grams from 3 to 6 characters.

At the end of optimization the program will save two files: `model.bin` and `model.vec`.

`model.vec` is a text file containing the word vectors, one per line. `model.bin` is a binary file containing the parameters of the model along with the dictionary and all hyper parameters.

The binary file can be used later to compute word vectors or to restart the optimization.

The following `fasttext(1)` command is equivalent



bigmeercat

被 317人关注，获得了178个喜欢

✚关注

ls /var/lib/dpkg/info/*.list -lht |less 查看包的安装日期*

基本信息

分类：机器学习

收录时间：2016-08-24 10:07:14

文章目录

- fasttext

```
# Skipgram model
./fasttext skipgram -input data.txt -output model

# CBOW model
./fasttext cbow -input data.txt -output model
```

Obtaining word vectors for out-of-vocabulary words

The previously trained model can be used to compute word vectors for out-of-vocabulary words.

```
print model['king'] # get the vector of the word 'king'
```

the following `fasttext(1)` command is equivalent:

```
echo "king" | ./fasttext print-vectors model.bin
```

This will output the vector of word `king` to the standard output.

Load pre-trained model

We can use `fasttext.load_model` to load pre-trained model:

```
model = fasttext.load_model('model.bin')
print model.words # list of words in dictionary
print model['king'] # get the vector of the word 'king'
```

Text classification

This package can also be used to train supervised text classifiers and load pre-trained classifier from fastText.

In order to train a text classifier using the method described in 2, we can use the following function:

```
classifier = fasttext.supervised('data.train.txt', 'model
```

equivalent as `fasttext(1)` command:

```
./fasttext supervised -input data.train.txt -output model
```

where `data.train.txt` is a text file containing a training sentence per line along with the labels. By default, we assume that labels are words that are prefixed by the string `__label__` .

We can specify the label prefix with the `label_prefix` param:

```
classifier = fasttext.supervised('data.train.txt', 'model
```

equivalent as `fasttext(1)` command:

```
./fasttext supervised -input data.train.txt -output model
```

This will output two files: `model.bin` and `model.vec` .

Once the model was trained, we can evaluate it by computing the precision at 1 (P@1) and the recall on a test set using `classifier.test` function:

```
result = classifier.test('test.txt')
print 'P@1:', result.precision
print 'R@1:', result.recall
print 'Number of examples:', result.nexamples
```

This will print the same output to stdout as:

```
./fasttext test model.bin test.txt
```

In order to obtain the most likely label for a list of text, we can use `classifer.predict` method:

```
texts = ['example very long text 1', 'example very longte
labels = classifier.predict(texts)
print labels

# Or with the probability
labels = classifier.predict_proba(texts)
print labels
```

We can specify `k` value to get the k-best labels from classifier:

```
labels = classifier.predict(texts, k=3)
print labels

# Or with the probability
labels = classifier.predict_proba(texts, k=3)
print labels
```

This interface is equivalent as `fasttext(1)` predict command. The same model with the same input set will have the same prediction.

API documentation

Skipgram model

Train & load skipgram model

```
model = fasttext.skipgram(params)
```

List of available `params` and their default value:

```
input_file      training file path (required)
output          output file path (required)
lr               learning rate [0.05]
lr_update_rate change the rate of updates for the learnin
dim              size of word vectors [100]
ws               size of the context window [5]
epoch            number of epochs [5]
min_count       minimal number of word occurences [5]
neg              number of negatives sampled [5]
word_ngrams     max length of word ngram [1]
loss             loss function {ns, hs, softmax} [ns]
bucket          number of buckets [2000000]
minn             min length of char ngram [3]
maxn             max length of char ngram [6]
thread          number of threads [12]
t                sampling threshold [0.0001]
silent          disable the log output from the C++ exten
encoding        specify input_file encoding [utf-8]
```

Example usage:

```
model = fasttext.skipgram('train.txt', 'model', lr=0.1, d
```

CBOW model

Train & load CBOW model

```
model = fasttext.cbow(params)
```

List of available `params` and their default value:

```
input_file      training file path (required)
output          output file path (required)
lr              learning rate [0.05]
lr_update_rate change the rate of updates for the learnin
dim             size of word vectors [100]
ws              size of the context window [5]
epoch           number of epochs [5]
min_count       minimal number of word occurences [5]
neg             number of negatives sampled [5]
word_ngrams     max length of word ngram [1]
loss            loss function {ns, hs, softmax} [ns]
bucket          number of buckets [2000000]
minn            min length of char ngram [3]
maxn            max length of char ngram [6]
thread          number of threads [12]
t               sampling threshold [0.0001]
silent          disable the log output from the C++ exten
encoding        specify input_file encoding [utf-8]
```

Example usage:

```
model = fasttext.cbow('train.txt', 'model', lr=0.1, dim=3
```

Load pre-trained model

File `.bin` that previously trained or generated by fastText can be loaded using this function

```
model = fasttext.load_model('model.bin', encoding='utf-8'
```

Attributes and methods for the model

Skipgram and CBOW model have the following atributes & methods

```
model.model_name        # Model name
model.words              # List of words in the dictionary
model.dim                # Size of word vector
model.ws                 # Size of context window
model.epoch              # Number of epochs
model.min_count         # Minimal number of word occurenc
model.neg                # Number of negative sampled
model.word_ngrams       # Max length of word ngram
model.loss_name         # Loss function name
model.bucket             # Number of buckets
model.minn               # Min length of char ngram
model.maxn               # Max length of char ngram
model.lr_update_rate    # Rate of updates for the learning
model.t                   # Value of sampling threshold
model.encoding          # Encoding of the model
model[word]              # Get the vector of specified wor
```

Supervised model

Train & load the classifier

```
classifier = fasttext.supervised(params)
```

List of available `params` and their default value:

```
input_file                      training file path
output                          output file path (
label_prefix                    label prefix ['__la
lr                              learning rate [0.1]
lr_update_rate                  change the rate of updates
dim                             size of word vecto
ws                              size of the contex
epoch                           number of epochs [5
min_count                       minimal number of w
neg                             number of negatives
word_ngrams                     max length of word
loss                            loss function {ns,
bucket                          number of buckets
minn                            min length of char
maxn                            max length of char
thread                          number of threads
t                               sampling threshold
silent                          disable the log out
encoding                        specify input_file
pretrained_vectors              pretrained word vectors (.
```

Example usage:

```
classifier = fasttext.supervised('train.txt', 'model', lab
                                 thread=4)
```

Load pre-trained classifier

File `.bin` that previously trained or generated by fastText can be loaded using this function.

```
./fasttext supervised -input train.txt -output classifier
```

```
classifier = fasttext.load_model('classifier.bin', label_p
```

Test classifier

This is equivalent as `fasttext(1)` test command. The test using the same model and test set will produce the same value for the precision at one and the number of examples.

```
result = classifier.test(params)

# Properties
result.precision # Precision at one
result.recall    # Recall at one
result.nexamples # Number of test examples
```

The param `k` is optional, and equal to `1` by default.

Predict the most-likely label of texts

This interface is equivalent as `fasttext(1)` predict command.

`texts` is an array of string

```
labels = classifier.predict(texts, k)

# Or with probability
labels = classifier.predict_proba(texts, k)
```

The param `k` is optional, and equal to `1` by default.

Attributes and methods for the classifier

Classifier have the following atributes & methods

```
classifier.labels                    # List of labels
classifier.label_prefix              # Prefix of the label
classifier.dim                       # Size of word vecto
classifier.ws                        # Size of context wi
classifier.epoch                     # Number of epochs
classifier.min_count                 # Minimal number of w
classifier.neg                       # Number of negative
classifier.word_ngrams               # Max length of word
classifier.loss_name                 # Loss function name
classifier.bucket                    # Number of buckets
classifier.minn                      # Min length of char
classifier.maxn                      # Max length of char
classifier.lr_update_rate            # Rate of updates for
classifier.t                         # Value of sampling
classifier.encoding                  # Encoding that used
classifier.test(filename, k)         # Test the classifier
classifier.predict(texts, k)         # Predict the most li
classifier.predict_proba(texts, k) # Predict the most lik
```

The param `k` for `classifier.test`, `classifier.predict` and `classifier.predict_proba` is optional, and equal to `1` by default.

References

Enriching Word Vectors with Subword Information

[1] P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov, *Enriching Word Vectors with Subword Information*

```
@article{bojanowski2016enriching,
  title={Enriching Word Vectors with Subword Information}
  author={Bojanowski, Piotr and Grave, Edouard and Joulin
  journal={arXiv preprint arXiv:1607.04606},
  year={2016}
}
```

Bag of Tricks for Efficient Text Classification

[2] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, *Bag of Tricks for Efficient Text Classification*

```
@article{joulin2016bag,
  title={Bag of Tricks for Efficient Text Classification}
  author={Joulin, Armand and Grave, Edouard and Bojanowsk
  journal={arXiv preprint arXiv:1607.01759},
  year={2016}
}
```

(* These authors contributed equally.)

Join the fastText community

- Facebook page: https://www.facebook.com/groups/1174547215919768
- Google group: https://groups.google.com/forum/#!forum/fasttext-library

相关教程- 更多教程

1、第五章无线攻击|ViolentPython中文版【Violent Python 中文版】
2、MySQL数据库(1)【《零基础学 python》（第二版）】
3、python3-cookbook2.0.0文档【Python Cookbook 3rd 中文版】
4、Tornado文档【Tornado 用户指南】
5、第二章app的模式|Django设计模式与最佳实践【Django 设计模式与最佳实践】
6、关于python调用zabbixapi接口的自动化实例【结合sa】【Python 实战-从菜鸟到大牛的进阶之路】
7、python调用zabbix的api接口添加主机、查询组、主【Python 实战-从菜鸟到大牛的进阶之路】
8、重定向(redirect)|Tornado概览【Tornado 概览】

相关主题- 发表话题

1、如何用 Python 和 Flask 建立部署一个 Facebook Messenger 机器人
2、中文版python资源全汇总
3、Prophet（先知）：Facebook大规模预报框架
4、Python里的黄金库，学会了你的工资至少翻一倍
5、浅析Python的Django框架中的Memcached
6、Linux 平台下 Python 脚本编程入门（二）
7、适用于 PHP 开发人员的 Python 基础知识
8、PyCharm 2017.1 正式发布：有更快的调试器

相关的项目 - 更多比较

机器学习 779 9.6k 1.5k

Awesome Tensorflow: 与Tensorflow相关的资源集合

A Awesome Tensorflow: 与Tensorflow相关的资源集合

机器学习 779 9.6k 1.5k

TensorFlow超酷专有资源大全

T TensorFlow超酷专有资源大全，收集了TensorFlow的实验、库和项目等等资源。

10.0 4.2 6天前

10.0 4.2 6天前

机器学习 653 8.8k 2k

一个工具包用于开发和比较强化学习算法

一个工具包用于开发和比较强化学习算法

Y

10.0 1.3 15天前