fastText一个库用于词表示的高效学习和句子分类

标签：proc ant 良好的 pes 场景 learning type 管道 lua

fastText

fastText 是 Facebook 开发的一个用于高效学习单词呈现以及语句分类的开源库。

要求

fastText 使用 C++11 特性，因此需要一个对 C++11 支持良好的编译器，可以使用：

- (gcc-4.6.3 或者更新版本) 或者 (clang-3.3 或者更新版本)

我们使用 Makefile 进行编译，因此需要 make 工具。为了运行单词相似度演示脚本，我们需要如下工具：

- python 2.6 or newer
- numpy & scipy

构建 fastText

使用如下命令来构建 fastText 库：

```
$ git clone git@github.com:facebookresearch/fastText.git
$ cd fastText
$ make
```

这将会为所有的类产生一堆文件，包括主二进制文件fasttext．如果你不打算用系统默认的编译器，在Makefile（CC 和 INCLUDES）的头部修改两个宏定义.

使用样例

这个包有两个主要功能：单词特征学习与文本分类．这都在以面两份论文[1] and [2]中有描述

单词特征学习

为了学习单词向量，就像[1]描述的那样：如下操作：

```
$ ./fasttext skipgram -input data.txt -output model
```

data.txt是一个训练文件，包含一些以utf-8编码的文本．默认的这些词向量将会划入字符(3致6个字符)帐目 g-grams . 最后的分析程序会保存为两个文件：model.bin 和　model.vec．　model.vec是文本文件包含单词向量，每个单词一行．model.bin是二进制文件包含字典模型参数与所有的其它参数．　这个二进制文件可以用于计算单词向量或重新分析。

第 2 段（可获 2.01 积分）

0  负愚価歸11个月前

从输出单词处获取单词向量

前期的训练模型可以从输出单词处计算词向量．假如你有一个文本文件queries.txt包含一些你想切分的单词向量，运用下面的命令：

分享档案 更多>

2017年12月16日 (493)

2017年12月15日 (982)

2017年12月14日 (1578)

2017年12月13日 (1256)

2017年12月12日 (998)

2017年12月11日 (1210)

2017年12月10日 (920)

2017年12月09日 (1245)

2017年12月08日 (1011)

2017年12月07日 (1095)

$ ./fasttext print-vectors model.bin < queries.txt

这会将单词向量输出到标准输出，一个向量一行．你也可以使用管道：

$ cat queries.txt | ./fasttext print-vectors model.bin

上面的脚本只是一个示例，为了更形像点运行：

$ ./word-vector-example.sh

第 3 段（可获 0.86 积分）


0负愚俪歸11个月前

这将会编译代码，下载数据，计算词向量，并可以测试那些由很少出现的词组成的数据集，测试它们的相似性[例如Thang 等等]．

文本分类

这个类库也可以用来监督文本分类训练，例如情绪分析．[2]里面描述可以用于训练文本分类, 使用：

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

train.txt是包含训练语句的文本文件，每行都带有标签，默认情况下，我们假设标签为单词，用前后加下划线的单词表示　如__label__．这个命令将会生成两个文件：model.bin 和　model.vec．一旦模型被训练，你可以评价它，用第一部分来测试计算它的精度：

第 4 段（可获 1.4 积分）


0负愚俪歸11个月前

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

为了获得一段文本最相似的标签，可以使用如下命令：

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

test.txt 包含一些文本用来根据每行进行分类。执行完毕将会输出每一行的近似标签。请看 classification-example.sh 来了解示例代码的使用场景。为了从论文 [2] 中重新生成结果，可以运行 classification-results.sh 脚本，这将下载所有的数据集并从表1中重新生成结果。

命令完整文档

```
The following arguments are mandatory:
  -input      training file path
  -output     output file path


The following arguments are optional:
  -lr         learning rate [0.05]
  -dim        size of word vectors [100]
  -ws         size of the context window [5]
  -epoch      number of epochs [5]
  -minCount   minimal number of word occurences [1]
  -neg        number of negatives sampled [5]
  -wordNgrams 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]
  -verbose    how often to print to stdout [1000]
  -t          sampling threshold [0.0001]
  -label      labels prefix [__label__]
```

第 5 段（可获 0.86 积分）

0 CY211个月前

参考资料

如果使用这些代码用于学习单词的呈现请引用 [1] ，如果用于文本分类请引用 [2]。

[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, Armand and Mikolov, Tomas},
  journal={arXiv preprint arXiv:1607.04606},
  year={2016}
}
```

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

第 6 段 (可获 0.63 积分)

0 CY211个月前

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

(* 这些作者贡献一样.)

加入 fastText 社区

- Facebook page: https://www.facebook.com/groups/1174547215919768
- Contact: egrave@fb.com, bojanowski@fb.com, ajoulin@fb.com, tmikolov@fb.com

fastText一个库用于词表示的高效学习和句子分类

fastText一个库用于词表示的高效学习和句子分类

-

fastText

fastText is a library for efficient learning of word representations and sentence classification.

Requirements

fastText builds on modern Mac OS and Linux distributions. Since it uses C++11 features, it requires a compiler with good C++11 support. These include :

- (gcc-4.6.3 or newer) or (clang-3.3 or newer)

Compilation is carried out using a Makefile, so you will need to have a working make. For the word-similarity evaluation script you will need:

- python 2.6 or newer
- numpy & scipy

Building fastText

In order to build fastText, use the following:

```
$ git clone https://github.com/facebookresearch/fastText.git
$ cd fastText
$ make
```

This will produce object files for all the classes as well as the main binary fasttext. If you do not plan on using the default system-wide compiler, update the two macros defined at the beginning of the Makefile (CC and INCLUDES).

Example use cases

This library has two main use cases: word representation learning and text classification. These were described in the two papers 1 and 2.

Word representation learning

In order to learn word vectors, as described in 1, do:

```
$ ./fasttext skipgram -input data.txt -output model
```

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.binis 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.

Obtaining word vectors for out-of-vocabulary words

The previously trained model can be used to compute word vectors for out-of-vocabulary words. Provided you have a text file queries.txt containing words for which you want to compute vectors, use the following command:

```
$ ./fasttext print-vectors model.bin < queries.txt
```

This will output word vectors to the standard output, one vector per line. This can also be used with pipes:

```
$ cat queries.txt | ./fasttext print-vectors model.bin
```

See the provided scripts for an example. For instance, running:

```
$ ./word-vector-example.sh
```

will compile the code, download data, compute word vectors and evaluate them on the rare words similarity dataset RW [Thang et al. 2013].

Text classification

This library can also be used to train supervised text classifiers, for instance for sentiment analysis. In order to train a text classifier using the method described in 2, use:

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

where 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__. This will output two files: model.binand model.vec. Once the model was trained, you can evaluate it by computing the precision and recall at k (P@k and R@k) on a test set using:

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

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

In order to obtain the k most likely labels for a piece of text, use:

```
$ ./fasttext predict model.bin test.txt k
```

where test.txt contains a piece of text to classify per line. Doing so will print to the standard output the k most likely labels for each line. The argument k is optional, and equal to 1 by default. See classification-example.sh for an example use case. In order to reproduce results from the paper 2, run classification-results.sh, this will download all the datasets and reproduce the results from Table 1.

If you want to compute vector representations of sentences or paragraphs, please use:

```
$ ./fasttext print-vectors model.bin < text.txt
```

This assumes that the text.txt file contains the paragraphs that you want to get vectors for. The program will output one vector representation per line in the file.

Full documentation

Invoke a command without arguments to list available arguments and their default values:

```
$ ./fasttext supervised
Empty input or output path.


The following arguments are mandatory:
  -input              training file path
```

```
  -output             output file path


The following arguments are optional:
  -lr                 learning rate [0.1]
  -lrUpdateRate       change the rate of updates for the learning rate [100]
  -dim                size of word vectors [100]
  -ws                 size of the context window [5]
  -epoch              number of epochs [5]
  -minCount           minimal number of word occurences [1]
  -minCountLabel      minimal number of label occurences [0]
  -neg                number of negatives sampled [5]
  -wordNgrams         max length of word ngram [1]
  -loss               loss function {ns, hs, softmax} [ns]
  -bucket             number of buckets [2000000]
  -minn               min length of char ngram [0]
  -maxn               max length of char ngram [0]
  -thread             number of threads [12]
  -t                  sampling threshold [0.0001]
  -label              labels prefix [__label__]
  -verbose            verbosity level [2]
  -pretrainedVectors  pretrained word vectors for supervised learning []
```

Defaults may vary by mode. (Word-representation modes skipgram and cbow use a default -minCount of 5.)

References

Please cite 1 if using this code for learning word representations or 2 if using for text classification.

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, Armand and Mikolov, Tomas},
  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 Bojanowski, Piotr and Mikolov, Tomas},
  journal={arXiv preprint arXiv:1607.01759},
  year={2016}
}
```

(* These authors contributed equally.)

Resources

You can find the preprocessed YFCC100M data used in [2]

at https://research.facebook.com/research/fasttext/

Join the fastText community

- Facebook page: https://www.facebook.com/groups/1174547215919768
- Google group: https://groups.google.com/forum/#!forum/fasttext-library
- Contact: egrave@fb.com, bojanowski@fb.com, ajoulin@fb.com, tmikolov@fb.com

See the CONTRIBUTING file for information about how to help out.

fastText一个库用于词表示的高效学习和句子分类

标签：proc ant 良好的 pes 场景 learning type 管道 lua

JD.COM 京东

1
2
3
4
5
6
7

雯娜诗秋装印花连衣裙冬加 ￥ 197.00

评论 一句话评论（0）

共0条

登录后才能评论！ 登录

友情链接

兰亭集智 国之画 百度统计 站长统计 阿里云 chrome插件 数安时代

迷上了代码！