项目实验报告

陈少泽　51194506005

一、实验目的

本次项目为 fasttext 模型，fasttext 是一种高效的词向量学习和句子分类模型。运行 GitHub 上的代码(https://github.com/facebookresearch/fastText/)，完成词向量的学习和文本分类任务。

二、实验环境和训练集

实验环境：

- 硬件：Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
- 系统环境：Red Hat Enterprise Linux Server release 7.5

使用的数据集合有：

- Wikipedia 数据集
  (http://mattmahoney.net/dc/enwik9.zip)
- the cooking section of stackexchange
  (https://dl.fbaipublicfiles.com/fasttext/data/cooking.stackexchange.tar.gz)
- 语言识别训练集
  (https://tatoeba.org/eng/downloads)

三、实验内容

1. fasttext 模型的词向量学习
2. 利用 fasttext 进行文本分类和语言识别

四、实验步骤

1. 项目参数：

字典参数	描述
-minCount	单词最少出现次数(默认值为 1)
-minCountLabel	标签最少出现的次数(默认值为 0)
-wordNgrams	字词段的最大字母长度(默认值为 1)
-bucket	默认值为 200000
-t	采样阈值(默认值为 0.0001)
-label	标签前缀(默认为__label__)

训练参数	描述
-lr	学习速率（默认值为 0.1)
-lrUpdateRate	改学习速率的更新速率（默认值为 100）
-dim	词向量的维度（默认值为 100）
-ws	窗口大小（默认值为 5）
-epoch	训练次数（默认值为 5）
-neg	负采样数量（默认值为 5）
-loss	损失函数{ns，hs，softmax}（默认 softmax）
-thread	CPU 线程数（默认值 12）
-pertreainedVectors	用于监督学习的预先训练的词向量
-saveOutput	是否将输出参数存储（默认值为 0）

2. 训练流程

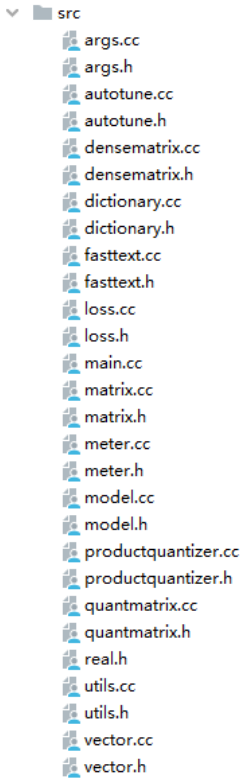

图 1

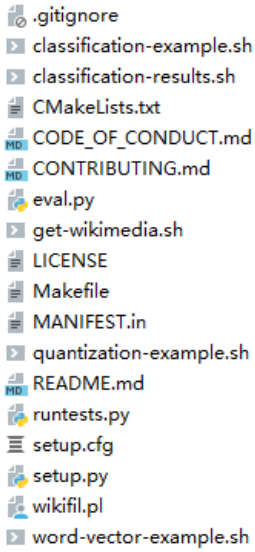

图 2

项目文件中分别给了文本分类的例子 classification-example.sh 和词向量训练的例子 word-vector-example.sh。src 文件夹中展示了各个功能的代码。下面介绍文本分类和词向量的训练流程。

Classification-example.sh:

```
myshuf() {
  perl -MList::Util=shuffle -e 'print shuffle(<>);' "$@";
}

normalize_text() {
  tr '[:upper:]' '[:lower:]' | sed -e 's/^/__label__/g' | \
    sed -e "s/'/ ' /g" -e 's/"//g' -e 's/\./ \. /g' -e 's/<br \/>/ /g' \
        -e 's/,/ , /g' -e 's/(/ ( /g' -e 's/)/ ) /g' -e 's/\!/ \! /g' \
        -e 's/\?/ \? /g' -e 's/\;/ /g' -e 's/\:/ /g' | tr -s " " | myshuf
}
```

文本分类任务中先将文本进行预处理，去除大小写差异和符号，只保留文本。最后对文本随机排序，打乱原有的顺序。

```
RESULTDIR=result
DATADIR=data

mkdir -p "${RESULTDIR}"
mkdir -p "${DATADIR}"

if [ ! -f "${DATADIR}/dbpedia.train" ]
then
  wget -c "https://github.com/le-
scientifique/torchDatasets/raw/master/dbpedia_csv.tar.gz" -O
"${DATADIR}/dbpedia_csv.tar.gz"
  tar -xzvf "${DATADIR}/dbpedia_csv.tar.gz" -C "${DATADIR}"
  cat "${DATADIR}/dbpedia_csv/train.csv" | normalize_text >
"${DATADIR}/dbpedia.train"
  cat "${DATADIR}/dbpedia_csv/test.csv" | normalize_text >
"${DATADIR}/dbpedia.test"
fi

make
```

创建数据目录和结果目录，下载数据。将数据分为训练集和测试集。用 make 命令编译 c++代码。

```
./fasttext supervised -input "${DATADIR}/dbpedia.train" -output
"${RESULTDIR}/dbpedia" -dim 10 -lr 0.1 -wordNgrams 2 -minCount 1 -bucket 10000000
-epoch 5 -thread 4

./fasttext test "${RESULTDIR}/dbpedia.bin" "${DATADIR}/dbpedia.test"

./fasttext predict "${RESULTDIR}/dbpedia.bin" "${DATADIR}/dbpedia.test" >
"${RESULTDIR}/dbpedia.test.predict"
```

选取合适的参数，如学习率、词向量维度、训练次数等。根据训练集输出训练结果。命令 ./fasttext test 对训练好的模型进行测试，使用精确率和召回率评价指标。

word-vector-example.sh：

```
mkdir -p "${RESULTDIR}"
mkdir -p "${DATADIR}"

if [ ! -f "${DATADIR}/fil9" ]
then
  wget -c http://mattmahoney.net/dc/enwik9.zip -P "${DATADIR}"
  unzip "${DATADIR}/enwik9.zip" -d "${DATADIR}"
  perl wikifil.pl "${DATADIR}/enwik9" > "${DATADIR}"/fil9
fi

if [ ! -f "${DATADIR}/rw/rw.txt" ]
then
  wget -c https://nlp.stanford.edu/~lmthang/morphoNLM/rw.zip -P "${DATADIR}"
  unzip "${DATADIR}/rw.zip" -d "${DATADIR}"
fi

make
```

先用 wget 命令下载 Wikipedia 数据集，用提供的 wikifil.pl 程序将 Wikipedia XML 转储过滤为仅包含小写字母和不连续空格的干净文本。所有其他字符都转换为空格。表格和图片删除，图表说明的文字部分保留。最后将链接转换为普通文本。同样使用 make 命令对 c++代码进行编译。

```
./fasttext skipgram -input "${DATADIR}"/fil9 -output "${RESULTDIR}"/fil9 -lr
0.025 -dim 100 \
  -ws 5 -epoch 1 -minCount 5 -neg 5 -loss ns -bucket 2000000 \
  -minn 3 -maxn 6 -thread 4 -t 1e-4 -lrUpdateRate 100

cut -f 1,2 "${DATADIR}"/rw/rw.txt | awk '{print tolower($0)}' | tr '\t' '\n' >
"${DATADIR}"/queries.txt

cat "${DATADIR}"/queries.txt | ./fasttext print-word-vectors
"${RESULTDIR}"/fil9.bin > "${RESULTDIR}"/vectors.txt

python eval.py -m "${RESULTDIR}"/vectors.txt -d "${DATADIR}"/rw/rw.txt
```

对预处理过的文本进行训练，使用的训练模型是 skip-gram。

五、实验结果

1. 文本分类：

使用 predict 对句子进行标签预测

```
(cszpy37) [sqli@hlogin01 fastText-0.9.1]$ ./fasttext predict-prob result/model_cooking.bin - 3
Which baking dish is best to bake a banana bread ?
__label__baking 0.545882 __label__bananas 0.235276 __label__bread 0.137298
Why not put knives in the dishwasher?
__label__knives 0.581434 __label__pressure-cooker 0.0818176 __label__equipment 0.079217
```

图 3 句子分类

```
(cszpy37) [sqli@hlogin01 fastText-0.9.1]$ ./fasttext test result/model_cooking.bin data/cooking.valid2
N       3000
P@1     0.587
R@1     0.254
```

图 4 分类的准确率和召回率

语言识别任务：

```
cszpy37) [sqli@hlogin01 data]$ head -n 10 all.txt
__label__spa Mary no consentirá que tomen el pelo a sus hermanos.
__label__eng I beg your pardon. I didn't know this was your seat.
__label__pes آیا به من اعتقاد دارید؟
__label__jpn 私は間違っていますか?
__label__rus Я уезжаю в Германию.
__label__eng Tom and Mary tried to protect their facesd.
__label__ber Tellid tceɣled?
__label__rus Том мог бы встретить вас в аэропорту.
__label__rus Да как ты посмел привести меня в мотель?
 label  eng Mary is waiting for someone to do that for her.
```

图 5

```
(cszpy37) [sqli@hlogin01 fastText-0.9.1]$ ./fasttext test result/langdetect.bin data/valid.txt
N       10000
P@1     0.984
R@1     0.984
```

图 6 语言识别准确率和召回率

2. 词向量学习

临近词向量查询：

```
Query word? asparagus
cabbage 0.744001
carrots 0.734136
vegetables 0.725723
artichokes 0.724955
tomato 0.722141
lettuce 0.716685
edible 0.714833
onions 0.713471
almonds 0.70904
potatoes 0.706339
```

图 7

```
Query word? pidgey
pidgeotto 0.804083
pidgeot 0.797755
pidge 0.766866
beedrill 0.689053
pok 0.670692
charizard 0.659327
raticate 0.656864
butterfree 0.641095
kakuna 0.637629
squirtle 0.637446
```

图 8

```
Query word? enviroment
enviromental 0.926588
enviro 0.725912
enviromission 0.660066
environ 0.62829
environment 0.606268
environmental 0.544775
environmentally 0.511572
degradation 0.505443
macroenvironmental 0.49483
cogeneration 0.492244
```

图 9

图 7 中，输入了菠菜得到了胡萝卜、卷心菜、蔬菜等相似词，fasttext 对语义的提取有着不错的效果。图 8 中，输入的是 Pokemon 动漫中的角色名，得到的前几

名答案都是该角色不同阶段的名字，对这些稀少词语的训练也有着较好的效果。图 9 中，输入了一个拼写错误的单词，可以看到 fasttext 模型依然捕捉到了单词的主要信息，进行了合理的匹配。虽然有一些影响，但是整体的预测方向是正确的。

单词类比：

```
Query triplet (A - B + C)? psx sony nintendo
gamecube 0.680144
snes 0.673629
nintendogs 0.666108
gba 0.6619
famicom 0.651852
nes 0.651063
sega 0.648645
playstation 0.643783
playstationjapan 0.635477
dreamcast 0.631424
```

图 10

```
Query triplet (A - B + C)? berlin germany france
paris 0.777354
rouen 0.635329
lille 0.614085
nantes 0.610571
grenoble 0.605473
bourges 0.585421
pavillon 0.583374
rennes 0.582461
strasbourg 0.58178
nanterre 0.580653
```

图 11

图 10，输入 psx ，sony，nintendo，模型认为 psx 是索尼的游戏手柄，因此 nintendo 任天堂类比的是 gamecube，这个类比也比较合理。图 11 中输入柏林、德国和法国得出巴黎，这个结果合理。