自然语言处理系列（3）：中文维基语料词向量训练

要通过计算机进行自然语言处理，首先就需要将这些文本数字化，目前用的最广泛的方法是词向量，根据训练使用算法的不同，目前主要有Word2Vec和GloVe两大方法，本文主要讲述通过这两个方法分别训练中文维基百科语料库的词向量。

1. [WechatIMG1zhanghua]()

一、获取并处理中文维基百科语料库

1.1 下载

中文维基百科语料库的下载链接为：https://dumps.wikimedia.org/zhwiki/，本试验下载的是最新的zhwiki-latest-pages-articles.xml.bz2。这个压缩包里面存的是标题、正文部分，该目录下还包括了其他类型的语料库，如仅包含标题，摘要等。

1.2 抽取内容

Wikipedia Extractor是一个开源的用于抽取维基百科语料库的工具，由python携程，通过这个工具可以很容易地从语料库中抽取相关内容。使用方法如下：

```
$ git clone https://github.com/attardi/wikiextractor.git wikiextractor
$ wikiextractor/WikiExtractor.py  -b 2000M -o zhwiki_extracted zhwiki-latest-pages-art
```

由于这个工具就是一个python脚本，因此无需安装，`-b`参数指对提取出来的内容进行切片后每个文件的大小，如果要将所有内容保存在同一个文件，那么就需要把这个参数设置地大一点，`-o`的参数指提取出来的文件放置的目录，抽取出来的文件的路径为`zhwiki_extract/AA/wiki_00`。更多的参数可参考其github主页的说明。

抽取后的内容格式为每篇文章被一对`<doc></doc>`包起来，而`<doc>`中的包含了属性有文章的id、url和title属性，如`<doc id="13" url="https://zh.wikipedia.org/wiki?curid=13" title="数学">`

1.3 繁简转换

由上一步提取出来的中文维基百科中的语料中既有繁体字也有简体字，这里需要将其统一变为简

体字，采用的工具也是开源的OpenCC转换器。安装使用方法如下：

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)" < /dev/null 2> /dev/null
$ brew install opencc
$ cd OpenCC
$ opencc -i /Users/HuaZhang/Desktop/zhwiki_extracted/AA/zhwiki_extract/AA/wiki_00 -o zhwiki_extract/zhs_wiki -c /Users/HuaZhang/OpenCC/data/config/t2s.json
```

其中`-i`表示输入文件路径，`-o`表示输出的文件，`-c`表示转换的配置文件，这里使用的繁体转简体的配置文件，OpenCC自带了一系列的转换配置文件，可以参考其github主页的说明。

1.4 去除标点

去除标点符号有两个问题需要解决，一个是像下面这种为了解决各地术语，名称不同的问题：

```
他的主要成就包括Emacs及後來的GNU Emacs，GNU C 編譯器及-{zh-hant:GNU 除錯器;zh-hans:GDB 调试器}-。
```

另外一个就是将所有标点符号替换成空字符，通过正则表达式均可解决这两个问题，下面是具体实现的python代码：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys
import re
import io

reload(sys)
sys.setdefaultencoding('utf-8')

def pre_process(input_file, output_file):
    multi_version = re.compile(ur'-\{.*?(zh-hans|zh-cn):([^;]*?)(;.*?)?\}-')
    punctuation = re.compile(u"[-~!@#$%^&*()_+`=\[\]\\\{\}\"|;':,./<>?·！ @#￥%……&* （）——+【】、；‘：“”，。、《》？ 「『」』]")
    with io.open(output_file, mode = 'w', encoding = 'utf-8') as outfile:
        with io.open(input_file, mode = 'r', encoding ='utf-8') as infile:
            for line in infile:
                line = multi_version.sub(ur'\2', line)
                line = punctuation.sub('', line.decode('utf8'))
                outfile.write(line)
```

```
if __name__ == '__main__':
    if len(sys.argv) != 3:
        print "Usage: python script.py input_file output_file"
        sys.exit()
    input_file, output_file = sys.argv[1], sys.argv[2]
    pre_process(input_file, output_file)
```

经过该步骤处理之后，得到了简体中文的纯净文本，如下所示：

> doc id13 urlhttpszhwikipediaorgwikicurid13 title数学
> 数学
> 数学是利用符号语言研究数量结构变化以及空间等概念的一门学科从某种角度看属于形式科学的一种数学透过抽象化和逻辑推理的使用由计数计算量度和对物体形状及运动的观察而产生数学家们拓展这些概念为了公式化新的猜想以及从选定的公理及定义中建立起严谨推导出的定理
> .....
> 数学奖通常和其他科学的奖项分开数学上最有名的奖为菲尔兹奖创立于1936年每四年颁奖一次它通常被认为是数学的诺贝尔奖另一个国际上主要的奖项为阿贝尔奖创立于2003年两者都颁奖于特定的工作主题包括数学新领域的创新或已成熟领域中未解决问题的解答著名的23个问题称为希尔伯特的23个问题于1900年由德国数学家大卫希尔伯特所提出这一连串的问题在数学家之间有著极高的名望且至少有九个问题已经被解答了出来另一新的七个重要问题称为千禧年大奖难题发表于2000年对其每一个问题的解答都有著一百万美元的奖金而当中只有一个问题黎曼猜想和希尔伯特的问题重复
> doc

1.5 jieba分词

下面需要对其进行分词并且整理成每行一篇文本的格式，从而方便后续的处理。

分词采用 python 的分词工具 jieba，通过 `pip install jieba` 安装即可。且将一篇文章分词后的结果存储在一行，由前面可知，每篇文章存储在一对 `<doc></doc>` 标签中，由于前面去掉了标点，所以现在变成了 `doc doc` ,所以只要判断当前行为doc时即可认为文章结束，从而开始在新的一行记录下一篇文章的分词结果。实现的python代码如下:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys
import io
import jieba

reload(sys)
```

```
sys.setdefaultencoding('utf-8')


def cut_words(input_file, output_file):
    count = 0
    with io.open(output_file, mode = 'w', encoding = 'utf-8') as outfile:
        with io.open(input_file, mode = 'r', encoding = 'utf-8') as infile:
            for line in infile:
                line = line.strip()
                if len(line) < 1:  # empty line
                    continue
                if line.startswith('doc'): # start or end of a passage
                    if line == 'doc': # end of a passage
                        outfile.write(u'\n')
                        count = count + 1
                        if(count % 1000 == 0):
                            print('%s articles were finished.......' %count)
                    continue
                for word in jieba.cut(line):
                    outfile.write(word + ' ')
    print('%s articles were finished.......' %count)

if __name__ == '__main__':
    if len(sys.argv) < 3:
        print "Usage: python script.py input_file output_file"
        sys.exit()
    input_file, output_file = sys.argv[1], sys.argv[2]
    cut_words(input_file, output_file)
```

二、通过Word2Vec训练词向量

Word2vec中包含了两种训练词向量的方法：Continuous Bag of Words(CBOW)和Skip-gram。CBOW的目标是根据上下文来预测当前词语的概率。Skip-gram刚好相反，根据当前词语来预测上下文的概率。这两种方法都利用人工神经网络作为它们的分类算法。起初，每个单词都是一个随机N维向量。训练时，该算法利用CBOW或者Skip-gram的方法获得了每个单词的最优向量。

最初 Google 开源的 Word2Vec 是用C来写的，后面陆续有了Python ， Java 等语言的版本，这里采用的是 Python 版本的 gensim。通过 gensim 提供的 API 可以比较容易地进行词向量的训练。gensim的建议通过`conda install gensim`安装

下面是对上面处理后的语料库进行训练的一个简单例子。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

```
import os, sys
import multiprocessing
import gensim

reload(sys)
sys.setdefaultencoding('utf-8')


def word2vec_train(input_file, output_file):
    sentences = gensim.models.word2vec.LineSentence(input_file)
    model = gensim.models.Word2Vec(sentences, size=300, min_count=10, sg=0, workers
=multiprocessing.cpu_count())
    model.save(output_file)
    model.wv.save_word2vec_format(output_file + '.vector', binary=True)


if __name__ == '__main__':
    if len(sys.argv) < 3:
        print "Usage: python script.py infile outfile"
        sys.exit()
    input_file, output_file = sys.argv[1], sys.argv[2]
    word2vec_train(input_file, output_file)
```

上面的训练过程首先将输入的文件转为 `gensim` 内部的 `LineSentence` 对象，要求输入的文件的格式为每行一篇文章，每篇文章的词语以空格隔开。

然后通过 `gensim.models.Word2Vec` 初始化一个 `Word2Vec` 模型， `size` 参数表示训练的向量的维数； `min_count` 表示忽略那些出现次数小于这个数值的词语，认为他们是没有意义的词语，一般的取值范围为（0，100）； `sg` 表示采用何种算法进行训练，取0时表示采用 `CBOW` 模型，取1表示采用 `skip-gram` 模型； `workers` 表示开多少个进程进行训练，采用多进程训练可以加快训练过程，这里开的进程数与CPU的核数相等。

假设我们训练好了一个语料库的词向量，当一些新的文章加入这个语料库时，如何训练这些新增的文章从而更新我们的语料库？将全部文章再进行一次训练显然是费时费力的，gensim提供了一种类似于“增量训练”的方法。即可在原来的model基础上仅对新增的文章进行训练。如下所示为一个简单的例子：

```
model = gensim.models.Word2Vec.load(exist_model)
model.train(new_sentences)
```

上面的代码先加载了一个已经训练好的词向量模型，然后再添加新的文章进行训练，同样新增的文章的格式也要满足每行一篇文章，每篇文章的词语通过空格分开的格式。这里需要注意的是加

载的模型只能 是通过`model.save()`存储的模型，从`model.save_word2vec_format()`恢复过来的模型只能用于查询.

三、使用词向量模型

训练好的词向量可以供后续的多项自然语言处理工作使用，下面是通过gensim加载训练好的词向量模型并进行查询的例子：

```
# 加载模型
import gensim
model = gensim.models.KeyedVectors.load_word2vec_format('/Users/HuaZhang/Desktop/zhwiki_extracted/output_word2vec.vector',binary = True)

# 词向量维度
len(model[u'黑格尔'])

# 相似度
model.similarity(u'叔本华',u'康德')
0.62428547493158093

# 找出相似度最高的词
words = model.most_similar(u"哈耶克")
for word in words:
    print word[0],word[1]

米塞斯 0.776977062225
叔本华 0.731572449207
黑格尔 0.723797023296
巴维克 0.723039865494
门格尔 0.719911754131
谢林 0.714867889881
波普尔 0.714080870152
马克思 0.711268663406
尼采 0.710071563721
博姆 0.705146193504

# 找出最不相关的词汇
print  model.doesnt_match(u"莎士比亚 卡夫卡 卢梭 爱因斯坦".split())
爱因斯坦
```