中山大學
SUN YAT-SEN UNIVERSITY

Sept 16, 2015
Version 2

TheB

用 Scrapy 及 Gensim 对哔哩哔哩弹幕网的标签进行 Word2Vec 语义分析

陈昊亮

https://chl.la/theb-2.pdf

目录

1. 项目简介

本项目，TheB，是一个课程项目。

1.1 项目目的

TheB 的目的在于运用现在信息检索技术，对哔哩哔哩弹幕网（bilibili.com）（下称 B 站）的标签/关键字进行语义分析，帮助用户寻找 B 站文化中的联系。

1.2 实现方式

项目实现分为三个部分，第一部分是使用爬虫，对 B 站的视频页面标签进行搜集；第二部分是使用语义分析，对搜集的标签进行分析；最后第三部分是用户界面，有一个界面应对用户的询问。

1.2.1 爬虫

本项目的爬虫使用 Scrapy[1]实现。Scrapy 是一个开源的爬虫框架。本项目选用 Scrapy 作

[1] http://scrapy.org/

为爬虫框架，主要基于它的以下几个优点：

- 开源
- 免费
- 文档丰富
- 功能足够

即便如此，由于 Scrapy 的配置文件较为分散，而文档又较为抽象，加上一开始又不是很熟悉 XPath，所以调试 Scrapy 所用的时间占了整个项目实践的大头。与第一版 TheB 靠蜘蛛爬行不同，第二版 TheB 改用顺序访问。因为 B 站视频网址序号连续，顺序访问效率高很多。

1.2.2 语义分析

本项目使用 Gensim[2]语义分析库。Gensim 是一个开源的语义分析库。本项目选用 Gensim 作为语义分析库主要基于它的一下几个优点：

- 开源
- 免费
- 文档丰富
- 功能强大
- 是 Python 库，降低整体项目开发代价

Gensim 的配置比较简单，运行快速，对于整个维基百科英文版[3]的 Word2Vec 语义分析，在我的个人电脑的虚拟机[4]上也只用了一个晚上。在服务器[5]上对 2878725fl（见后文）跑 8192 维的分析一小时内即可解决。

1.2.3 用户界面

项目的实现时基于 Python 的，所以用户界面也使用 Python 内置的 Tkinter 库实现。

2. 项目实现

本章将分三部分，分别介绍爬虫、语义分析及用户界面的实现。

[2] http://radimrehurek.com/gensim/index.html
[3] https://dumps.wikimedia.org/enwiki/20150515/enwiki-20150515-pages-articles.xml.bz2
[4] i5-4750，RAM 8G，HDD 5400Rev
[5] 双路 E5-2620，RAM96G，SSD

2.1 Scrapy

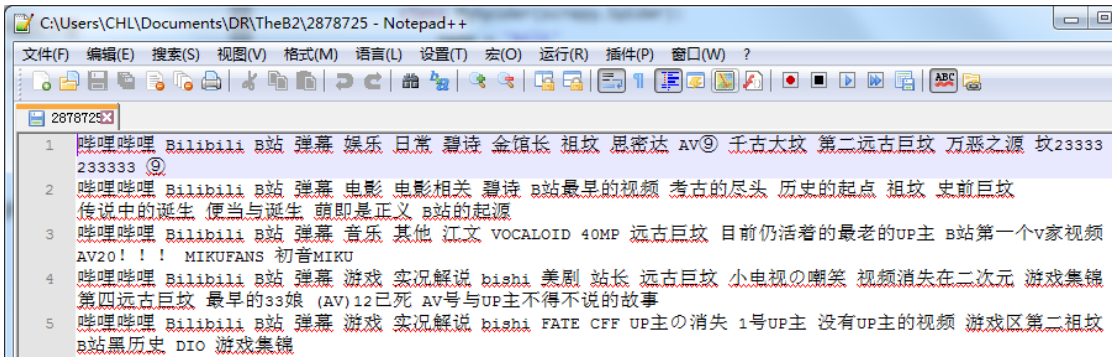
以下是 Scrapy 中 Spider 的详细定义。

```
import codecs
import scrapy
from b.items import CHL
import re

class MySpider(scrapy.Spider):
    name = 'bili'
    allowed_domains = ['www.bilibili.com']
    amy = ['http://www.bilibili.com/']
    for i in range(7,2878725):
        amy.append("http://www.bilibili.com/video/av"+str(i))
    start_urls = amy

    def parse(self, response):
        item = CHL();
        try:
            item['tag'] = re-
sponse.xpath('//meta[@name=\'keywords\']/@content').extract()[0]
            thestr = item['tag'].replace(',',' ') + '\n'
            with codecs.open('2878725', 'a', 'utf-8') as f:
                f.write(thestr)
        except IndexError:
            pass
        return item
```

输出的文件 2878725 如下



一行是一个视频的所有 Tag，以空格分隔，方便 Gensim 处理。

由于有很多链接的视频失效了，或者是因为需要会员权限才能观看，它们的 Tag 都是一致的 B 站默认 Tag，所以需要过滤掉以获得更准确的模型。过滤前的结果有 2 699 531 行，315M；过滤后的结果有 1 720 808 行，共 188M。

2.2 Gensim

以下是使用了 Gensim 库作语义分析的 Python 程序。输入文件是 input，输出文件是 w2v 和 w2v.trim。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import gensim, logging, codecs
class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname

    def __iter__(self):
        for line in codecs.open(self.dirname, 'r', 'utf-8'):
            yield line.split()

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)
sentences = MySentences('input')
model = gensim.models.Word2Vec(sentences, size=8192, workers=24,
min_count=3)
model.save('w2v')
model.init_sims(replace=True)
model.save('w2v.trim')
```

由于结果 numpy 数组很大，所以输出时还会输出若干.npy 文件保存数据。

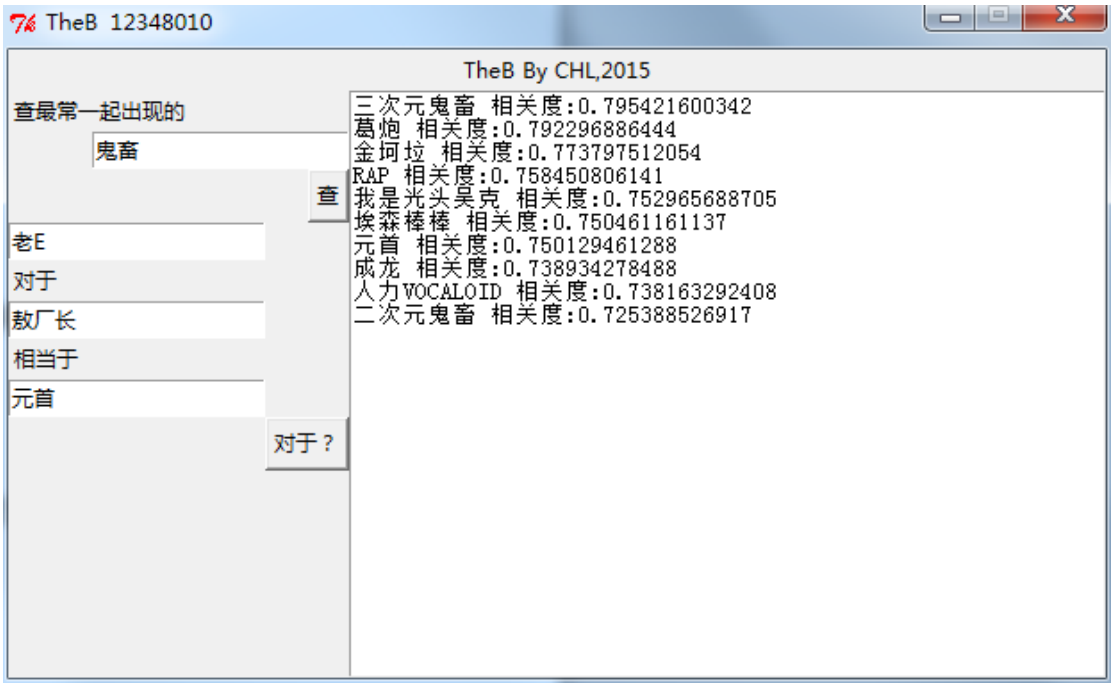
2.3 Tkinter

Tkinter 程序十分简单。但由于定义框架的语句重复多且冗长，就不再这里贴了。

3. 成果

最终产品 TheB，拥有两个功能。可以查找与关键字最接近的主题，以及可以做类比查询。

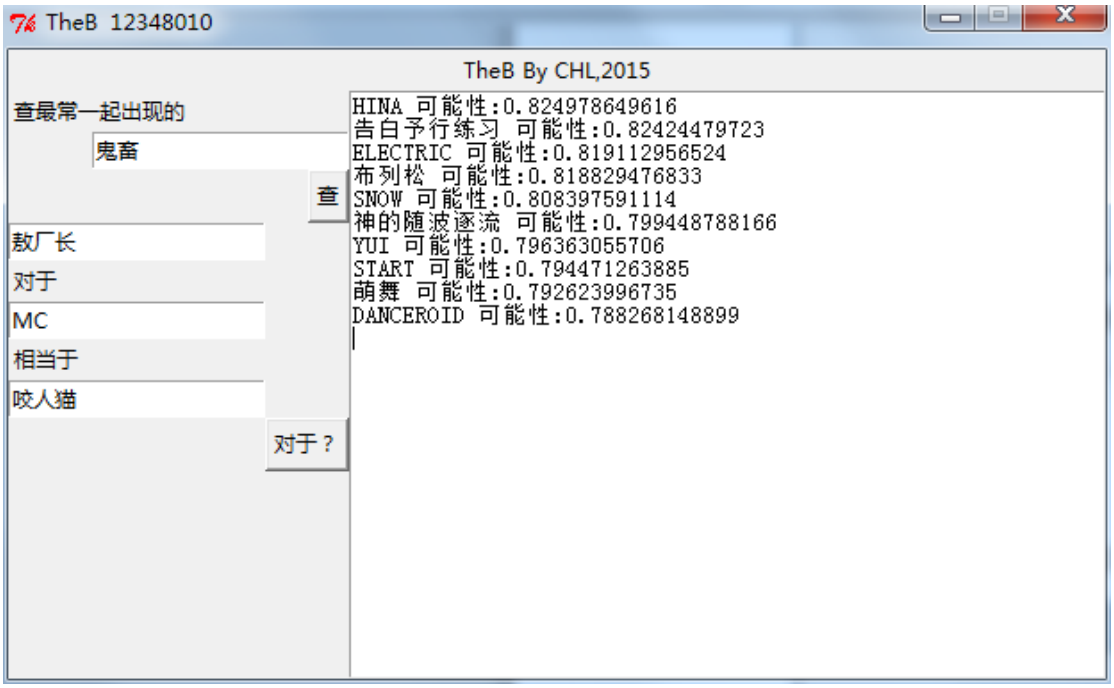
3.1 相近主题查询



上述结果显示，与“鬼畜”这个关键字最接近的 10 个主题分别是“三次元鬼畜”、“葛炮”、“金坷垃”等等。

3.2 类比查询

类比查询的意思如下，“man”对于“boy”相当于“woman”对于“girl”。



上述结果显示，“敖厂长”（一个游戏视频作者）对于“MC”（一款游戏）相当于“咬人猫”（一个舞蹈视频作者）对于“HINA”（另一个舞蹈视频作者）或者“告白予行练习”（一

支舞蹈）。

4. 总结

再一次修订文档时，得益于心的爬虫架构，整个 B 站的公开 Tag 都能爬下来了。

Word2Vec 需要大量的训练数据才能达到较高的准确率，而在使用 Gensim 对维基百科进行 W2V 分析时，还发现 size 参数，即向量的维度对模型的性能有较大影响。在所以所有数据都有作了 400、1024、8192 三个 size 的处理。

使用 TheB 能发现 B 站许多自己不曾关注的特点。比如在查找“诸葛亮”的相关结果时，会有“你妈挂树上啦”这一项。经深入了解发现，“你妈挂树上啦”是一个受大众欢迎的鬼畜视频作者。又比如类比查询中的例子，第一个结果 HINA 其实是一个有不少受众的舞者，但是我却从来未听过。如果不是 TheB，我不会了解到很多我会感兴趣，但是我平时没有遇上的视频作者、视频分类。

综上所述，TheB 达到了预期效果，相信在完善后能带给用户意想不到的好处。