BosonNLP HTTP API Documentation

Release 1.0

上海玻森数据科技有限公司

December 25, 2015

CONTENTS

Tip: 本文档的 PDF 版本可以从 这里 下载。

为了提供更加稳定可靠的服务，我们需要对每一个 API 请求进行鉴权验证。所有接口在调用时，需要在 HTTP 请求的头部（header）里设置一个名为 `X-Token` 的字段。在本文档中，我们将使用 `YOUR_API_TOKEN` 作为 `X-Token` 的值进行说明。

Note: 如果您不知道您的 `Token` 是什么，请和我们联系。

CHAPTER

ONE

快速上手指南

1.1 使用入门

1.1.1 介绍

BosonNLP 引擎以 REST API 的方式提供服务，任何编程语言都可以轻松使用。

在正式开始前，您需要首先 注册玻森账号。完成后，您将在 控制台 的底部看到您的 API Token （密钥），该密钥将用于身份验证。

1.1.2 Hello World

这里将以一个简单的情感分析任务为例，介绍 BosonNLP 的使用。

我们从 cURL 开始。

打开一个命令行窗口并输入以下命令（不包含 `$` ），将 `YOUR_API_TOKEN` 替换为您注册后获得的 **API** 密钥。

```
$ curl http://api.bosonnlp.com/sentiment/analysis \
    -X POST -H "Content-Type: application/json" -H "X-Token: YOUR_API_TOKEN" \
    --data "[\" 你要明白，自由思考比畅所欲言更重要。\", \" 大公司最大的困扰，就是无法准确测量每个员工的贡献
```

```
[[0.6620696791981926, 0.3379303208018074], [0.348186633910877, 0.651813366089123]]
```

以上这段代码使用了玻森的情感分析 引擎，传入了两段短文本内容进行分析。返回的内容为 `json` 格式，情感分析结果分别为 `非负面` 和 `负面` 概率组成的列表。

1.1.3 HTTP Header 详解

在 cURL 命令中加入 `-i` 参数，会看到类似下面的结果。

```
$ curl http://api.bosonnlp.com/sentiment/analysis -i \
    -X POST -H "Content-Type: application/json" -H "X-Token: YOUR_API_TOKEN"\
    --data "[\" 你要明白，自由思考比畅所欲言更重要。\", \" 大公司最大的困扰，就是无法准确测量每个员工的贡献
```

```
HTTP/1.1 200 OK
Date: Sun, 04 May 2014 08:21:01 GMT
Content-Type: application/json
Content-Length: 6
Connection: keep-alive
X-Rate-Limit-Limit: 100
X-Rate-Limit-Remaining: 97
X-Rate-Limit-Reset: 1399192200
X-Count-Limit-Limit: 500
X-Count-Limit-Remaining: 480
X-Count-Limit-Reset: 1399219200
Server: nginx/1.5.11
X-Request-Id: 0ae45f04-701d-48d8-a84e-d08f18e489ef

[[0.6620696791981926, 0.3379303208018074], [0.348186633910877, 0.651813366089123]]
```

在返回的 HTTP Header 当中，有一些很有趣的内容。和你想的一样，BosonNLP 的返回内容为 JSON 格式，因此 `Content-Type` 是 `application/json` 。

`X-` 开头的是自定义 HTTP 头，由 BosonNLP 生成，其中的信息非常有用。例如，

- `X-Request-Id` 是对每个请求生成的唯一 ID，用于在引擎内部跟踪请求。
- `X-Count-Limit-Remaining` 是当前可用的调用次数。
- `X-Count-Limit-Reset` 是调用次数重置的时间。当前时间窗口中调用次数用尽时，等待到这里指定的时间才可以恢复使用。

这里可以看到关于调用限制的详细内容。

1.1.4 使用 Python SDK

如果您使用 Python 语言，建议通过 SDK 的方式使用 BosonNLP。

BosonNLP Python SDK 是由 BOSON 官方支持的开发者工具包，提供了对 REST 接口的简化封装。

最简便的安装方式是通过 pip 。

```
$ pip install -U bosonnlp
```

安装成功后，编写以下 Python 脚本，并保存为 `sentiment.py` 。在代码中，将 `YOUR_API_TOKEN` 更换为您的 API 密钥。

```
# -*- coding: utf-8 -*-
from __future__ import print_function, unicode_literals
from bosonnlp import BosonNLP

nlp = BosonNLP('YOUR_API_TOKEN')
print(nlp.sentiment('宝马中国召回 1418 辆进口 X5 汽车'))
```

运行。

```
$ python sentiment.py
[[0.2279116935591028, 0.7720883064408972]]
```

通过 SDK 调用，对以上内容的情感分析结果为 负面概率较大。

详细的 BosonNLP Python SDK 文档请看 这里 。

1.1.5 Node.js SDK（第三方）

如果您是 Node.js 开发者，可以使用由 liwenzhu 开发和维护的 BosonNLP Node SDK。

1.1.6 继续阅读

恭喜！您已经学会了使用 BosonNLP 进行情感分析 。BosonNLP 其他的单文本分析工具调用方式也很类似。

您可以继续查看 BosonNLP 每个引擎的详细介绍，进一步了解每个引擎的特性和调用方法。

- 单文本分析
- 多文本分析
- 应用场景示例

CHAPTER

TWO

单文本分析

2.1 情感分析

2.1.1 介绍

本接口将文本的情感分为负面和非负面两类。

本引擎用微博、新闻、汽车、餐饮等不同行业语料进行标注和机器学习，调用时请通过 URL 参数选择特定的模型，以获得最佳的情感判断准确率。

URL `http://api.bosonnlp.com/sentiment/analysis`

HTTP Method `POST`

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP query string 参数 传递模型名选择用特定行业语料进行训练的模型。可选参数，默认为 `general` 。

模型名	行业	URL
general	通用	http://api.bosonnlp.com/sentiment/analysis
auto	汽车	http://api.bosonnlp.com/sentiment/analysis?auto
kitchen	厨具	http://api.bosonnlp.com/sentiment/analysis?kitchen
food	餐饮	http://api.bosonnlp.com/sentiment/analysis?food
news	新闻	http://api.bosonnlp.com/sentiment/analysis?news
weibo	微博	http://api.bosonnlp.com/sentiment/analysis?weibo

HTTP 请求 Body JSON 格式的需要做情感判定的文本或者文本组成的列表。比如：`["\u4ed6\u662f\u4e2a\u50bb\u903c", "\u7f8e\u597d\u7684\u4e16\u754c"]`

Note: 我们限定了一次传入的文章数目不能超过 100 篇。

HTTP 返回 Body JSON 格式的 [double, double] 类型组成的列表。每个元素分别表示请求的列表对应位置的文本的情感判断结果；第一个值为非负面概率，第二个值为负面概率，两个值相加和为 1。

2.1.2 CURL 调用示例

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\"\u4ed6\u662f\u4e2a\u50bb\u903c\", \"\u7f8e\u597d\u7684\u4e16\u754c\"]" \
    http://api.bosonnlp.com/sentiment/analysis
```

2.1.3 Python 调用示例

```
# -*- encoding: utf-8 -*-
import json
import requests


SENTIMENT_URL = 'http://api.bosonnlp.com/sentiment/analysis'
# 注意：在测试时请更换为您的 API token。
headers = {'X-Token': 'YOUR_API_TOKEN'}

s = ['他是个傻逼', '美好的世界']
data = json.dumps(s)
resp = requests.post(SENTIMENT_URL, headers=headers, data=data.encode('utf-8'))

print resp.text
```

2.1.4 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals

from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')
```

```
s = ['他是个傻逼', '美好的世界']

result = nlp.sentiment(s)

print result
```

详细的 Python SDK 情感分析文档请看 这里 。

2.1.5 PHP 调用示例

本示例使用 PHP 的 cURL 扩展 调用 BosonNLP 的 HTTP 接口。

示例里的两条文本是某家餐馆的点评网品论，所以我们传递 food URL 参数来选择针对餐饮行业优化的模型。

```
<?php
$API_TOKEN = "YOUR_API_TOKEN";
$SENTIMENT_URL = 'http://api.bosonnlp.com/sentiment/analysis?food';
$data = array(' 这家味道还不错', ' 菜品太少了而且还不新鲜');

$ch = curl_init();
curl_setopt_array($ch, array(
  CURLOPT_URL => $SENTIMENT_URL,
  CURLOPT_HTTPHEADER => array(
   "Accept:application/json",
   "Content-Type: application/json",
   "X-Token: $API_TOKEN",
  ),
  CURLOPT_POST => true,
  CURLOPT_POSTFIELDS => json_encode($data),
  CURLOPT_RETURNTRANSFER => true,
));

$result = curl_exec($ch);
var_dump(json_decode($result));

curl_close($ch);
```

运行

```
$ php sentiment.php
array(2) {
  [0]=>
  array(2) {
```

```
    [0]=>
    float(0.99943896320364)
    [1]=>
    float(0.00056103679636458)
  }
  [1]=>
  array(2) {
    [0]=>
    float(0.00017390916533533)
    [1]=>
    float(0.99982609083466)
  }
}
```

PHP cURL Class 是一个 PHP cURL 扩展的面向对象封装，提供更加易于理解和使用的 API。上面的示例代码使用 PHP cURL Class 写出来是这样：

```php
<?php
require 'vendor/autoload.php';

$API_TOKEN = "YOUR_API_TOKEN";
$SENTIMENT_URL = 'http://api.bosonnlp.com/sentiment/analysis?food';
$data = array(' 这家味道还不错', ' 菜品太少了而且还不新鲜');

use \Curl\Curl;

$curl = new Curl();
$curl->setHeader('Accept', 'application/json');
$curl->setHeader('Content-Type', 'application/json');
$curl->setHeader('X-Token', "$API_TOKEN");

$curl->post($SENTIMENT_URL, json_encode($data));

var_dump($curl->response);
```

2.1.6 Java 调用示例

下面是 Java 的 Unirest 库 (Github) 调用 Bosondata 的 Sentiment API 的示例。

Tip: 完整的示例代码可以从 这里 下载。

```java
package net.bosondata.sentiment_api_example;

import org.json.JSONArray;
import org.json.JSONException;
```

```
import com.mashape.unirest.http.HttpResponse;
import com.mashape.unirest.http.JsonNode;
import com.mashape.unirest.http.Unirest;
import com.mashape.unirest.http.exceptions.UnirestException;

public class SentimentApiExample
{
    public static final String SENTIMENT_URL =
        "http://api.bosonnlp.com/sentiment/analysis";

    public static void main(String[] args) throws JSONException, UnirestException,
                                                  java.io.IOException
    {
        String body = new JSONArray(new String[]{" 他是个傻逼", " 美好的世界"}).toString();
        HttpResponse<JsonNode> jsonResponse = Unirest.post(SENTIMENT_URL)
            .header("Accept", "application/json")
            .header("X-Token", "YOUR_API_TOKEN")
            .body(body)
            .asJson();

        System.out.println(jsonResponse.getBody());

        // Unirest starts a background event loop and your Java
        // application won't be able to exit until you manually
        // shutdown all the threads
        Unirest.shutdown();
    }
}
```

构建

```
$ mvn clean compile assembly:single
```

运行

```
$ java -cp target/sentiment-api-example-1.0-SNAPSHOT-jar-with-dependencies.jar \
    net.bosondata.sentiment_api_example.SentimentApiExample
```

应用场景示例

请看找出最负面的消费者评价

2.2 命名实体识别

命名实体识别（NER）是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等。命名实体识别是信息提取、问答系统、句法分析、机器翻译等应用领域的重要基础工具，作为结构化信息提取的重要步骤。

在 BosonNLP NER 中，我们将识别以下类别的实体：

时间	time
地点	location
人名	person_name
组织名	org_name
公司名	company_name
产品名	product_name
职位	job_title

URL `http://api.bosonnlp.com/ner/analysis`

HTTP Method `POST`

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP query string 参数

sensitivity 可选参数，用于调节准确率与召回率之间的平衡，取值为 1 （覆盖更多实体）到 5 （更准确）之间的整数，默认为 3 。

HTTP 请求 Body JSON 格式的需要做命名实体识别的文本或者文本组成的列表。比如：

```
["\u6210\u90fd\u5546\u62a5\u8bb0\u8005 \u59da\u6c38\u5fe0"]
```

Note: 我们限定了一次传入的文章数目不能超过 100 篇。

HTTP 返回 Body JSON 格式的实体识别引擎返回的结果。

key	type	说明
word	list	分词结果
tag	list	词性标注结果
entity	list	命名实体结果

其中命名实体结果为一个三元组：(s, t, entity_type) ，表示 word[s:t] 的内容为类型 entity_type 的实体。

2.2.1 CURL 调用示例

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\"\u6210\u90fd\u5546\u62a5\u8bb0\u8005 \u59da\u6c38\u5fe0\"]" \
    http://api.bosonnlp.com/ner/analysis
[{"tag": ["ns", "n", "n", "nr"],
  "word": [" 成都", " 商报", " 记者", " 姚永忠"],
  "entity": [[0, 2, "product_name"], [3, 4, "person_name"]]}]
```

2.2.2 Python 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals
import json
import requests


NER_URL = 'http://api.bosonnlp.com/ner/analysis'


s = ['对于该小孩是不是郑尚金的孩子，目前已做亲子鉴定，结果还没出来，'
     '纪检部门仍在调查之中。成都商报记者 姚永忠']
data = json.dumps(s)
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(NER_URL, headers=headers, data=data)


for item in resp.json():
    for entity in item['entity']:
        print(''.join(item['word'][entity[0]:entity[1]]), entity[2])
```

运行

```
$ python ner_api_example.py
郑尚金 person_name
成都商报 product_name
姚永忠 person_name
```

2.2.3 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals

from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token
nlp = BosonNLP('YOUR_API_TOKEN')
s = ['对于该小孩是不是郑尚金的孩子，目前已做亲子鉴定，结果还没出来，'
     '纪检部门仍在调查之中。成都商报记者 姚永忠']
result = nlp.ner(s)[0]
words = result['word']
entities = result['entity']


for entity in entities:
    print(''.join(words[entity[0]:entity[1]]), entity[2])
```

详细的 Python SDK 命名实体识别文档请看 这里 。

2.3 依存文法分析

依存文法分析核心思想为将一个线性描写的句子表述为成员之间的搭配与驱动关系。

依存文法分析引擎的依存具有如下 23 种关系：

名称	解释	举例
ROOT	核心词	警察 * 打击 * 犯罪。
SBJ	主语成分	*警察 * 打击犯罪。
OBJ	宾语成分	警察打击 * 犯罪 *。
PU	标点符号	你好 *!*
TMP	时间成分	*昨天下午 * 下雨了。
LOC	位置成分	我 * 在北京 * 开会。
MNR	方式成分	我 * 以最快的速度 * 冲向了终点。
POBJ	介宾成分	他 * 对客人 * 很热情。
PMOD	介词修饰	这个产品 * 直 * 到今天才完成。
NMOD	名词修饰	这是一个 * 大 * 错误。
VMOD	动词修饰	我 * 狠狠地 * 打 * 了 * 他。
VRD	动结式（第二动词为第一动词结果）	福建省 * 涌现出 * 大批人才。
DEG	连接词“的”结构	*我 * 的妈妈是超人。
DEV	“地”结构	他 * 狠狠 * 地看我一眼。
LC	位置词结构	我在 * 书房 * 里吃饭。
M	量词结构	我有 * 一 * 只小猪。
AMOD	副词修饰	一批 * 大 * 中企业折戟上海。
PRN	括号成分	北京（首都）很大。
VC	动词“是”修饰	我把你 * 看做 * 是妹妹。
COOR	并列关系	希望能 * 贯彻 * *执行 * 该方针
CS	从属连词成分	如果 * 可行 *，我们进行推广。
PMOD	介词修饰	*直 * 到今天我都记得。
DEC	关系从句“的”	这是 * 以前不曾遇到过 * 的情况。

URL `http://api.bosonnlp.com/depparser/analysis`

HTTP Method `POST`

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 请求 Body JSON 格式的需要做依存文法的分析的文本或者文本组成的列表。比如：`["\u6211\u4ee5\u6700\u5feb \u7684\u901f \u5ea6\u5403\u4e86\u5348\u996d"]`

Note: 我们限定了一次传入的文章数目不能超过 100 篇。

HTTP 返回 Body JSON 格式的依存文法分析引擎返回的结果。

2.3.1 CURL 调用示例

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\"\u6211\u4ee5\u6700\u5feb\u7684\u901f\u5ea6\u5403\u4e86\u5348\u996d\"]" \
    http://api.bosonnlp.com/depparser/analysis
[{"role": ["SBJ", "MNR", "VMOD", "DEC", "NMOD", "POBJ", "ROOT", "VMOD", "OBJ"],
  "head": [6, 6, 3, 4, 5, 1, -1, 6, 6],
  "tag": ["PN", "P", "AD", "VA", "DEC", "NN", "VV", "AS", "NN"],
  "word": ["\u6211", "\u4ee5", "\u6700", "\u5feb", "\u7684", "\u901f\u5ea6",
           "\u5403", "\u4e86", "\u5348\u996d"]}]
```

2.3.2 Python 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals
import json
import requests


DEPPARSER_URL = 'http://api.bosonnlp.com/depparser/analysis'


s = ['我以最快的速度吃了午饭']
data = json.dumps(s)
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(DEPPARSER_URL, headers=headers, data=data.encode('utf-8'))


for item in resp.json():
    print(' '.join(item['word']))
    print(' '.join(item['tag']))
    print(item['head'])
    print(' '.join(item['role']))
```

运行

```
$ python depparser_api_example.py
我 以 最 快 的 速度 吃 了 午饭
PN P AD VA DEC NN VV AS NN
[6, 6, 3, 4, 5, 1, -1, 6, 6]
SBJ MNR VMOD DEC NMOD POBJ ROOT VMOD OBJ
```

2.3.3 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from bosonnlp import BosonNLP
# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')

s = ['我以最快的速度吃了午饭']

result = nlp.depparser(s)

print(' '.join(result[0]['word']))
print(' '.join(result[0]['tag']))
print(result[0]['head'])
print(' '.join(result[0]['role']))
```

详细的 Python SDK 依存文法分析文档请看 这里 。

2.4 关键词提取

关键词作为一个对文本常用的概括，可以被应用于关键词云计算等应用上。BosonNLP 的关键词提取引擎可以将文本自动进行关键词分析，给出每个词语相应的权重。

关键词提取引擎的调用很简单，只需要将文本上传到分析服务器，引擎会自动返回每个词语的权重。

Note: 注意这些权重的平方和为 1。

URL `http://api.bosonnlp.com/keywords/analysis`

HTTP Method `POST`

HTTP Header

`Content-Type` `application/json`

`Accept` `application/json`

`X-Token` `YOUR_API_TOKEN` （需要替换成您自己的 Token）

HTTP query string 参数

`top_k` 可选的参数，默认值为 100。

`segmented` 可选的参数，表明传入的数据已进行了分词，则引擎不会再对内容进行分词处理。

HTTP 请求 Body JSON 格式的需要做关键词提取的文本。比如："\u75c5\u6bd2\u5f0f\u5a92\u4f53\u7f51\u7ad9\uff1a\u8ba9\u65b0\u95fb\u8fc5\u901f\u8513\u5ef6"

HTTP 返回 Body JSON 格式的引擎提取出的关键词相应的权重和关键词组成的列表。

2.4.1 CURL 调用示例

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "\"\u75c5\u6bd2\u5f0f\u5a92\u4f53\u7f51\u7ad9\uff1a\u8ba9\u65b0\u95fb\u8fc5\u901f\u8
    http://api.bosonnlp.com/keywords/analysis?top_k=2
[[0.4580507649282757, "\u8513\u5ef6"], [0.44467176143180404, "\u75c5\u6bd2"]]
```

2.4.2 Python 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals
import json
import requests


KEYWORDS_URL = 'http://api.bosonnlp.com/keywords/analysis'


text = '病毒式媒体网站：让新闻迅速蔓延'
params = {'top_k': 10}
data = json.dumps(text)
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(KEYWORDS_URL, headers=headers, params=params, data=data.encode('utf-8'))


for weight, word in resp.json():
    print(weight, word)
```

运行

```
$ python keywords_api_example.py
0.458050764928 蔓延
0.444671761432 病毒
```

```
0.377757036473 迅速
0.345847781395 网站
0.341803722579 媒体
0.315811136856 式
0.305528321034 新闻
0.142237269898 让
```

2.4.3 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token
nlp = BosonNLP('YOUR_API_TOKEN')
s = '病毒式媒体网站：让新闻迅速蔓延'
result = nlp.extract_keywords(s, top_k=10)


for weight, word in result:
    print(weight, word)
```

详细的 Python SDK 关键词提取文档请看 这里 。

2.5 新闻分类

本接口将新闻文本归类到预设的 14 个分类当中。

预设的分类如下：

编号	分类
0	体育
1	教育
2	财经
3	社会
4	娱乐
5	军事
6	国内
7	科技
8	互联网
9	房产
10	国际
11	女人
12	汽车
13	游戏

URL `http://api.bosonnlp.com/classify/analysis`

HTTP Method `POST`

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 请求 Body JSON 格式的需要做分类的新闻文本或者文本组成的列表。比如：

```
["\u4fc4\u5426\u51b3\u5b89\u7406\u4f1a\u8c34\u8d23\u53d9\u519b\u6218\u673a\u7a7a\u88ad\u9
 "\u9093\u7d2b\u68cb\u8c08\u7537\u53cb\u6797\u5ba5\u5609\uff1a\u6211\u89c9\u5f97\u6211\u6
 "Facebook\u6536\u8d2d\u5370\u5ea6\u521d\u521b\u516c\u53f8"]
```

Note: 我们限定了一次传入的文章数目不能超过 100 篇。

HTTP 返回 Body JSON 格式的分类编号（0 到 13 之间的数字）组成的列表。比如：

```
[5, 4, 8]
```

2.5.1 CURL 调用示例

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\"Facebook\u6536\u8d2d\u5370\u5ea6\u521d\u521b\u516c\u53f8\"]" \
```

```
    http://api.bosonnlp.com/classify/analysis
[8]
```

2.5.2 Python 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals
import json
import requests


CLASSIFY_URL = 'http://api.bosonnlp.com/classify/analysis'


s = [
    '俄否决安理会谴责叙军战机空袭阿勒颇平民',
    '邓紫棋谈男友林宥嘉：我觉得我比他唱得好',
    'Facebook 收购印度初创公司',
]

data = json.dumps(s)
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(CLASSIFY_URL, headers=headers, data=data.encode('utf-8'))


# shoud print [5, 4, 8]
print(resp.text)
```

2.5.3 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from bosonnlp import BosonNLP
# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')

s = [
    '俄否决安理会谴责叙军战机空袭阿勒颇平民',
    '邓紫棋谈男友林宥嘉：我觉得我比他唱得好',
    'Facebook 收购印度初创公司',
]

result = nlp.classify(s)

print result
```

详细的 Python SDK 新闻分类文档请看 这里 。

2.6 语义联想

该接口返回与输入词语最相近的 top_k 个词，最大值可设定为 100 。

URL http://api.bosonnlp.com/suggest/analysis

HTTP Method POST

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP query string 参数

top_k 可选的参数，默认值为 10。

HTTP 请求 Body JSON 格式的需要做语义联想的词。比如："\u7c89\u4e1d"

HTTP 返回 Body JSON 格式的列表。

Note: 如果输入的单词没有在我们学习的词库当中出现，则会返回一个空的 JSON 列表。

2.6.1 CURL 调用示例

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "\"\u7c89\u4e1d\"" \
    http://api.bosonnlp.com/suggest/analysis?top_k=2
[[0.99999999999999967, "\u7c89\u4e1d/n"], [0.48602467961311008, "\u8111\u6b8b\u7c89/n"]]
```

2.6.2 Python 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals
import json
import requests
```

```
SUGGEST_URL = 'http://api.bosonnlp.com/suggest/analysis'


term = '粉丝'
params = {'top_k': 10}
data = json.dumps(term)
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(SUGGEST_URL, headers=headers, params=params, data=data.encode('utf-8'))


for item in resp.json():
    print(item[0], item[1])
```

运行

```
$ python suggest_api_example.py
1.0 粉丝
0.416940319812 脑残粉
0.406041416995 歌迷
0.372752082893 粉
0.353264590233 博友
0.352079384235 听众
0.347602450534 后援
0.34479180559 乐迷
0.318045441765 博主
0.308489327951 10￥2000
```

2.6.3 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals

from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')

term = '粉丝'

result = nlp.suggest(term, top_k=10)

for item in result:
    print(item[0], item[1])
```

详细的 Python SDK 语义联想文档请看 这里 。

2.7 分词与词性标注

2.7.1 概述

因为中文的自然语言书写对于不同的词之间不会采用显示分隔符（如空格）进行分割，在大多数自然语言问题当中，分词都作为最基础的步骤。词性用来描述一个词在上下文中的作用，而词性标注就是识别这些词的词性，以确定其在上下文中的作用。一般情况下，词性标注是建立在分词基础上的另一个自然语言处理的基础步骤。为了适应 BosonNLP 自然语言处理的需要，BosonNLP 采用将分词和词性标注联合枚举的方法，实现了这一套分词和词性标注系统，并通过开放 API 接口的形式提供给其他开发者使用。

BosonNLP 的分词和词性标注都是基于序列标注实现的，以词为单位对句子进行词边界和词性的标注即发挥了基于字符串匹配方法切分速度快、效率高等特点，又可以结合上下文识别生词、自动消除歧义，同时避免由于分词错误造成词性标注错误的级联放大。

BosonNLP 分词和词性标注系统完全是自主实现的，在原有算法和语料的基础上，又加入了一些优化：

- 加入了对 url、email 等特殊词的识别
- 对词性标签进行调整和优化，实现了更细的标签划分（22 个大类，69 个标签）
- 对训练语料进行修正
- 加入繁简转化，可以处理繁体中文或者繁简混合的中文句子

BosonNLP 分词和词性标注系统还提供了多种分词选项，以满足不同开发者的需求：

- 空格保留选项
- 新词枚举强度选项
- 繁简转换选项
- 特殊字符转换选项

性能测试：

- 在人民日报测试集上测试结果：

	正确率	召回率	F1 值
分词	0.976725	0.981847	0.979279
分词和词性标注	0.954014	0.959017	0.956509

- 为了适应 BosonNLP 自然语言处理的需要，我们还准备了一个新的分词语料库，包括近两年的新闻、微博、点评等各个分类的数据。这个数据集上面会出现很多近几年出现的新词，一些不规范的网络术语，错别字等，因而处理难度更大。在这个数据集上面的测试结果如下：

	正确率	召回率	F1 值
分词	0.969493	0.974508	0.971994
分词和词性标注	0.946201	0.951096	0.948642

2.7.2 调用说明

各个标签对应词性见词性标注说明。

URL `http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0&&special_char_`

Parameters

`space_mode` （空格保留选项）

value	说明
0	不保留空格 (default)
1	连续出现的多个空格只保留一个
2	保留所有空格，不改变原文本的空格
3	对于英文单词间的空格不保留，中文词之间的空格连续多次出现只保留一个

`oov_level` （新词枚举强度选项）

value	说明
0	不枚举新词，只有在词典中出现的词才会出现在分词结果中
1-4	允许出现新词，从 1 到 4 出现新词的可能性依次增大
default	3 (正常水平)

`t2s` （繁简转换选项）

value	说明
0	关闭繁简转换，保留原文本（default）
1	将所有繁体中文转化成简体中文

`special_char_conv` （特殊字符转换选项）

value	说明
0	不进行特殊字符转换，保留原文本（default）
1	进行特殊字符转换，将”\n”, “\r”, “\t” 分别转换成”_Enter_”, “_Enter_” , “_Tab_”

HTTP Method POST

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 请求 Body JSON 格式的需要做分词与词性标注的文本或者文本组成的列表。比如：

```
"\u8fd9\u4e2a\u4e16\u754c\u597d\u590d\u6742"
```

Note: 我们限定了一次传入的文章数目不能超过 100 篇。

HTTP 返回 Body JSON 格式的分词与词性标注结果。

key	type	说明
word	list	分词结果
tag	list	词性标注结果

2.7.3 CURL 调用示例

不同的空格保留选项（space_mode）：

```
$ curl -X POST \
     -H "Content-Type: application/json" \
     -H "Accept: application/json" \
     -H "X-Token: YOUR_API_TOKEN" \
     --data "\" 人民法院案件受理制度改革　下月起法院将有案必立 \"" \
     'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0'
[{"tag": ["nl", "n", "n", "n", "n", "t", "f", "n", "d", "vyou", "n", "d", "v"], "word

$ curl -X POST \
     -H "Content-Type: application/json" \
     -H "Accept: application/json" \
     -H "X-Token: YOUR_API_TOKEN" \
     --data "\" 人民法院案件受理制度改革　下月起法院将有案必立 \"" \
     'http://api.bosonnlp.com/tag/analysis?space_mode=1&oov_level=3&t2s=0'
[{"tag": ["nl", "n", "n", "n", "n", "w", "t", "f", "n", "d", "vyou", "n", "d", "v"],

$ curl -X POST \
     -H "Content-Type: application/json" \
     -H "Accept: application/json" \
     -H "X-Token: YOUR_API_TOKEN" \
```

```
    --data "\" 人民法院案件受理制度改革　下月起法院将有案必立 \"" \
    'http://api.bosonnlp.com/tag/analysis?space_mode=2&oov_level=3&t2s=0'
[{"tag": ["nl", "n", "n", "n", "n", "w", "t", "f", "n", "d", "vyou", "n", "d", "v"],
```

不同的新词枚举强度选项（oov_level）：

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\" 亚投行意向创始成员国确定为 57 个 \",\"“流量贵”频被吐槽 \"]" \
    'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=0&t2s=0'
[{"tag": ["ns", "v", "n", "n", "vi", "n", "v", "v", "m", "q"], "word": [" 亚", " 投",

$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\" 亚投行意向创始成员国确定为 57 个 \",\"“流量贵”频被吐槽 \"]" \
    'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=1&t2s=0'
[{"tag": ["ns", "n", "n", "vi", "n", "v", "v", "m", "q"], "word": [" 亚", " 投行", " 意

$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\" 亚投行意向创始成员国确定为 57 个 \",\"“流量贵”频被吐槽 \"]" \
    'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0'
[{"tag": ["n", "n", "vi", "n", "v", "v", "m", "q"], "word": [" 亚投行", " 意向", " 创始

$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "[\" 亚投行意向创始成员国确定为 57 个 \",\"“流量贵”频被吐槽 \"]" \
    'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=4&t2s=0'
[{"tag": ["n", "n", "vi", "n", "v", "v", "m", "q"], "word": [" 亚投行", " 意向", " 创始
```

不同的繁简转换选项（t2s）：

```
$ curl -X POST \
    -H "Content-Type: application/json" \
    -H "Accept: application/json" \
    -H "X-Token: YOUR_API_TOKEN" \
    --data "\" 臺灣沿用傳統漢字，稱之為正體字 \"" \
    'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0'
[{"tag": ["ns", "v", "n", "nz", "wd", "v", "n"], "word": [" 臺灣", " 沿用", " 傳統", "
```

```
$ curl -X POST \
     -H "Content-Type: application/json" \
     -H "Accept: application/json" \
     -H "X-Token: YOUR_API_TOKEN" \
     --data "\" 臺灣沿用傳統漢字，稱之為正體字 \"" \
     'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=1'
[{"tag": ["ns", "v", "n", "n", "wd", "v", "n"], "word": [" 台湾", " 沿用", " 传统", " 
```

不同的特殊字符转换选项（special_char_conv）：

```
$ curl -X POST \
     -H "Content-Type: application/json" \
     -H "Accept: application/json" \
     -H "X-Token: YOUR_API_TOKEN" \
     --data "[\" 亚投行 意向创始成员国确定为 57 个 \n\",\"“流量贵”频被吐槽 \n\"]" \
     'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0&special_cha
[{"tag": ["n", "n", "vi", "n", "v", "v", "m", "q", "w"], "word": [" 亚投行", " 意向",

$ curl -X POST \
     -H "Content-Type: application/json" \
     -H "Accept: application/json" \
     -H "X-Token: YOUR_API_TOKEN" \
     --data "[\" 亚投行 意向创始成员国确定为 57 个 \n\",\"“流量贵”频被吐槽 \n\"]" \
     'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0&special_cha
[{"tag": ["n", "n", "vi", "n", "v", "v", "m", "q", "w"], "word": [" 亚投行", " 意向",
```

2.7.4 Python 调用示例

```python
# -*- encoding: utf-8 -*-
from __future__ import print_function, unicode_literals

import json
import requests


TAG_URL = 'http://api.bosonnlp.com/tag/analysis'
# 如果某个选项采用默认设置，可以在 TAG_URL 中省略，完整的 TAG_URL 如下：
# 'http://api.bosonnlp.com/tag/analysis?space_mode=0&oov_level=3&t2s=0&special_char_conv=0'
# 修改 space_mode 选项为 1
# TAG_URL = \
#   'http://api.bosonnlp.com/tag/analysis?space_mode=1'
# 修改 oov_level 选项为 1
# TAG_URL = \
#    'http://api.bosonnlp.com/tag/analysis?oov_level=1'
# 修改 t2s 选项为 1
# TAG_URL= \
```

```
#     'http://api.bosonnlp.com/tag/analysis?t2s=1'
# 修改 special_char_conv 选项为 1
# TAG_URL= \
# 'http://api.bosonnlp.com/tag/analysis?special_char_conv=1'

s = ['亚投行意向创始成员国确定为 57 个', '"流量贵"频被吐槽']
data = json.dumps(s)
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(TAG_URL, headers=headers, data=data.encode('utf-8'))


for d in resp.json():
    print(' '.join(['%s/%s' % it for it in zip(d['word'], d['tag'])]))
```

运行

```
$ python tag_api_example.py
亚投行/n 意向/n 创始/vi 成员国/n 确定/v 为/v 57/m 个/q
"/wyz 流量/n 贵/a "/wyy 频/d 被/pbei 吐槽/v
```

2.7.5 Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals

from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')

s = ['亚投行意向创始成员国确定为 57 个', '"流量贵"频被吐槽']

result = nlp.tag(s)
# 完整的参数调用格式如下：
# result = nlp.tag(s, space_mode=0, oov_level=3, t2s=0, special_char_conv=0)
# 修改 space_mode 选项为 1, 如下：
# result = nlp.tag(s, space_mode=1, oov_level=3, t2s=0, special_char_conv=0)
# 修改 oov_level 选项为 1, 如下：
# result = nlp.tag(s, space_mode=0, oov_level=1, t2s=0, special_char_conv=0)
# 修改 t2s 选项为 1, 如下：
# result = nlp.tag(s, space_mode=0, oov_level=3, t2s=1, special_char_conv=0)
# 修改特殊字符转换选项为 1, 如下：
# result = nlp.tag(s, space_mode=0, oov_level=3, t2s=0, special_char_conv=1)
```

```
for d in result:
    print(' '.join(['%s/%s' % it for it in zip(d['word'], d['tag'])]))
```

详细的 Python SDK 分词与词性标注文档请看 这里 。

2.8 时间转换

2.8.1 介绍

将中文时间描述转换为三种标准的时间格式的时间字符串：1) 时间点（timestamp，表示某一具体时间时间描述）；2) 时间量（timedelta，表示时间的增量的时间描述）；3) 时间区间（timespan，大于一天的有具体起始和结束时间点的时间描述），以方便处理。在转换时以尽量保留原始时间描述中所含信息为原则。

URL `http://api.bosonnlp.com/time/analysis`

HTTP Method `POST`, `GET`

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP query string 参数

pattern 必选参数，需要转换的中文时间描述。

basetime 可选的参数，时间描述时的基准时间戳。如果不传，使用服务器当前的 GMT+8 时间。

HTTP 请求 Body 空。

HTTP 返回 Body 返回数据为一个 JSON 格式的字典，时间格式类型字段中包含一种下表中的返回值，并在 type 标记中返回相应的时间格式类型。

时间格式类型	type 标记	数据格式	说明
时间点 (timestamp)	timestamp	string	时间点，ISO8601 格式的时间字符串
时间量 (timedelta)	timedelta	string	时间量，格式为"x day,HH:MM:SS"或"HH:MM:SS"的字符串
时间区间 (timespan)	timespan_0	list	表示时间点组成的时间区间结果，格式为[timestamp, timestamp] 表示时间区间的起始和结束时间
时间区间 (timespan)	timespan_1	list	时间区间结果，格式为[timedelta, timedelta]，表示时间区间的起始和结束时间
N/A	“”	string	不能识别，返回空字符串

2.8.2 CURL 调用示例

```
$ curl -X POST \
    -H "X-Token: YOUR_API_TOKEN" \
    http://api.bosonnlp.com/time/analysis?pattern=2013 年二月二十八日下午四点三十分二十九秒
    {"timestamp": "2013-02-28 16:30:29", "type": "timestamp"}
```

2.8.3 Python 调用示例

```
# -*- coding: utf-8 -*-

from __future__ import print_function, unicode_literals

import time

import requests

TIME_URL = 'http://api.bosonnlp.com/time/analysis'

pattern = '2013 年二月二十八日下午四点三十分二十九秒'
current_time = int(time.time())
params = {'pattern': pattern, 'basetime': current_time}
headers = {'X-Token': 'YOUR_API_TOKEN'}
resp = requests.post(TIME_URL, headers=headers, params=params)

print(resp.json())
```

运行

```
$ python time_api_example.py
{u'timestamp': u'2013-02-28 16:30:29', u'type': u'timestamp'}
```

2.8.4 Python SDK 调用示例

```
# -*- coding: utf-8 -*-

from __future__ import print_function, unicode_literals

import datetime
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
```

```
nlp = BosonNLP('YOUR_API_TOKEN')

result = nlp.convert_time(
    "2013 年二月二十八日下午四点三十分二十九秒",
    datetime.datetime.today())

print(result)
```

详细的 Python SDK 时间转换文档请看 这里 。

2.9 新闻摘要

2.9.1 概述

由于现今网络的发展，信息获取变的十分简单和方便。随之而来的弊端之一就是巨量的信息无法快速有效的处理以便后续使用。特别在新闻语料中，常出现大量重复、多余或者不重要信息的情况。对此，较直观的一种解决办法是对新闻做摘要，减少信息长度，即新闻摘要。

总的来说，新闻摘要是指，在给定的字数限制范围内，摘取单篇或多篇新闻原文中的句子，来代表该篇或多篇新闻的大意和中心思想。在做摘要的监督性和非监督性机器学习方法中，前者往往会产生重复语句的摘要。因此很多衍生出的监督性模型会通过增加句间关系来解决该问题，但也同时增加了复杂度。非监督性方法中，最广泛应用的是 MMR (Maximal Marginal Relevance) 和基于图形的 Text Rank 模型，前者在多样性上优于后者。

BosonNLP 的单篇新闻摘要系统是基于 MMR 模型自主实现的，在同时考量信息的相关性和多样性的情况下，保证了摘要结果的准确性和完整性，其复杂度相对较低。可涉及的新闻内容是开放的。

新闻摘要还可作为其他扩展应用的基础，因此开放 API 接口提供给其他开发者使用。现有的摘要系统提供 4 个输入选项：

- 新闻标题
- 新闻正文
- 字数限制
- 是否为严格字数限制

2.9.2 调用说明

API 连接 `http://api.bosonnlp.com/summary/analysis`

参数

参数名字	类型	说明
字数限制 (percentage)	float or int 0.3（default）	p <= 1，则认为是摘要字数与原文总字数的百分比； p > 1，则认为是最终摘要的具体字数
(not_exceed) 0（default） 1	boolean	是否开启严格字数限制开关 不开启，摘要字数大于等于字数限制以确保句子完整 开启，摘要字数小于等于字数限制

数据

数据	类型	说明
新闻标题（title）	unicode	新闻主标题，可不填写（即为空）
新闻正文（source）	unicode	新闻正文，不包括新闻主标题

HTTP Method `POST`

HTTP 请求 Body JSON 格式的需要做摘要的单篇文本。

Note: 单篇新闻字数不能超过 1 万字。

HTTP 返回 Body JSON 格式的摘要结果格式是一行一句的字符串。

2.9.3 Python 调用示例

```
# -*- coding: utf-8 -*-

from __future__ import unicode_literals
import json
import requests


SUMMARY_URL = 'http://api.bosonnlp.com/summary/analysis'
# 注意：在测试时请更换为您的 API token。
headers = {'X-Token': 'YOUR_API_TOKEN'}

source = {
    'not_exceed': 0,
    'percentage': 0.2,
```

```
    'title': '',
    'content': (
        '腾讯科技讯（刘亚澜）10 月 22 日消息，'
        '前优酷土豆技术副总裁黄冬已于日前正式加盟芒果 TV，出任 CTO 一职。'
        '资料显示，黄冬历任土豆网技术副总裁、优酷土豆集团产品技术副总裁等职务，'
        '曾主持设计、运营过优酷土豆多个大型高容量产品和系统。'
        '此番加入芒果 TV 或与芒果 TV 计划自主研发智能硬件 OS 有关。'
        '今年 3 月，芒果 TV 对外公布其全平台日均独立用户突破 3000 万，日均 VV 突破 1 亿，'
        '但挥之不去的是业内对其技术能力能否匹配发展速度的质疑，'
        '亟须招揽技术人才提升整体技术能力。'
        '芒果 TV 是国内互联网电视七大牌照方之一，之前采取的是“封闭模式”与硬件厂商预装合作，'
        '而现在是“开放下载”+“厂商预装”。'
        '黄冬在加盟土豆网之前曾是国内 FreeBSD（开源 OS）社区发起者之一，'
        '是研究并使用开源 OS 的技术专家，离开优酷土豆集团后其加盟果壳电子，'
        '涉足智能硬件行业，将开源 OS 与硬件结合，创办魔豆智能路由器。'
        '未来黄冬可能会整合其在开源 OS、智能硬件上的经验，结合芒果的牌照及资源优势，'
        '在智能硬件或 OS 领域发力。'
        '公开信息显示，芒果 TV 在今年 6 月对外宣布完成 A 轮 5 亿人民币融资，估值 70 亿。'
        '据芒果 TV 控股方芒果传媒的消息人士透露，芒果 TV 即将启动 B 轮融资。')
}

resp = requests.post(
    SUMMARY_URL,
    headers=headers,
    data=json.dumps(source).encode('utf-8'))

print json.loads(resp.text)
```

运行

```
$ python summary_api_example.py
腾讯科技讯（刘亚澜）10 月 22 日消息，前优酷土豆技术副总裁黄冬已于日前正式加盟芒果 TV，出任 CTO 一职。
未来黄冬可能会整合其在开源 OS、智能硬件上的经验，结合芒果的牌照及资源优势，在智能硬件或 OS 领域发力。
据芒果 TV 控股方芒果传媒的消息人士透露，芒果 TV 即将启动 B 轮融资。
```

2.9.4 Python SDK 调用示例

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
```

```
nlp = BosonNLP('YOUR_API_TOKEN')


content = (
    '腾讯科技讯（刘亚澜）10 月 22 日消息，'
    '前优酷土豆技术副总裁黄冬已于日前正式加盟芒果 TV，出任 CTO 一职。'
    '资料显示，黄冬历任土豆网技术副总裁、优酷土豆集团产品技术副总裁等职务，'
    '曾主持设计、运营过优酷土豆多个大型高容量产品和系统。'
    '此番加入芒果 TV 或与芒果 TV 计划自主研发智能硬件 OS 有关。'
    '今年 3 月，芒果 TV 对外公布其全平台日均独立用户突破 3000 万，日均 VV 突破 1 亿，'
    '但挥之不去的是业内对其技术能力能否匹配发展速度的质疑，'
    '亟须招揽技术人才提升整体技术能力。'
    '芒果 TV 是国内互联网电视七大牌照方之一，之前采取的是"封闭模式"与硬件厂商预装合作，'
    '而现在是"开放下载"+"厂商预装"。'
    '黄冬在加盟土豆网之前曾是国内 FreeBSD（开源 OS）社区发起者之一，'
    '是研究并使用开源 OS 的技术专家，离开优酷土豆集团后其加盟果壳电子，'
    '涉足智能硬件行业，将开源 OS 与硬件结合，创办魔豆智能路由器。'
    '未来黄冬可能会整合其在开源 OS、智能硬件上的经验，结合芒果的牌照及资源优势，'
    '在智能硬件或 OS 领域发力。'
    '公开信息显示，芒果 TV 在今年 6 月对外宣布完成 A 轮 5 亿人民币融资，估值 70 亿。'
)

result = nlp.summary('', content)

# 完整的参数调用如下：
# result = nlp.summary(title, content, word_limit=0.3, no_exceed=0)
# 修改标题为该文章标题，如下：
# title = '前优酷土豆技术副总裁黄冬加盟芒果 TV 任 CTO'
# result = nlp.summary(title, content, word_limit=0.3, no_exceed=0)
# 修改 word_limit 选项为百分之二十，如下：
# result = nlp.summary(title, content, word_limit=0.2, no_exceed=0)
# 修改 word_limit 选项为具体字数 200 时，如下：
# result = nlp.summary(title, content, word_limit=200, no_exceed=0)
# 修改 no_exceed 选项为严格不超出字数限制时，如下：
# result = nlp.summary(title, content, word_limit=0.3, no_exceed=1)

print result
```

CHAPTER

THREE

多文本分析

3.1 文本聚类引擎

由于自然语言书写文档的多样性，同样内容的文章往往可以采用不同的词句和语法构成进行书写。人对于文档的理解通常在话题的层面，而并非单一的文档或发贴。这使得我们希望机器自动能够对给定的文本进行话题聚类，将语义上相似的文章归为一类，方便人的浏览查看，可进行话题级的统计分析。比如下面两条微博：

- # 新闻追踪 #：【冀中星被移送检察院审查起诉】首都机场公安分局对冀中星爆炸案侦查终结，目前已移送朝阳检察院审查起诉。7 月 20 日 18 时 24 分，冀中星在首都机场 T3 航站楼 B 口外引爆自制炸药。案发当天除冀中星左手腕因被炸截肢外，无其他人伤亡。7 月 29 日，冀中星因涉嫌爆炸罪被批捕。http://t.cn/zQHjr0S
- # 豫广微新闻 #【首都机场爆炸案嫌犯冀中星移送检方审查起诉】据报道，首都机场公安分局对冀中星爆炸案侦查终结，目前已移送朝阳检察院审查起诉。7 月 20 日，山东籍男子冀中星在首都机场 T3 航站楼引爆自制炸药，案发当天除冀中星左手腕因被炸截肢外，无其他人伤亡

报道的同一个事件，虽然书写形式有所区别，文章标题也不一样，但由于语义的相似性，仍然可以被话题聚类引擎识别出来。目前 BosonNLP 采用的话题聚类算法为无监督算法，即不需要人工数据标注。

该引擎的使用需要以下三个步骤：

1. 上传需要聚类的文本数据；
2. 调用分析引擎（异步）；
3. 从服务器取得聚类的结果。

以下我们将按照上述的三个步骤来介绍引擎的调用。

3.1.1 上传数据

URL `http://api.bosonnlp.com/cluster/push/TASK_ID`

TASK_ID 是任务的 ID，用于唯一识别该聚类任务，可由字母和数字组成。

Warning: 对于每次聚类任务应使用不同的 TASK_ID ，以免导致混淆。

HTTP Method POST

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 请求 Body JSON 格式的列表。每个元素为上传到聚类引擎的文档编号和内容组成的 JSON 对象，必须包含两个字段：

1. **_id** 文档的编号，可以采用整数或字符串
2. **text** 文档的内容，原始的文章信息

对于一个聚类任务，建议可上传 50 条以上，20 万条以下内容进行分析。

Note: 注意，在文本数量较大时，请分批上传数据。每批最多上传不超过 100 条文本。

Java 调用代码示例

```
HttpResponse<JsonNode> clusterPushResponse = Unirest
    .post("http://api.bosonnlp.com/cluster/push/TASK_ID")
    .header("Accept", "application/json")
    .header("Content-Type", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .body(YOUR_JSON_TEXT)
    .asJson();
```

3.1.2 调用分析

一旦所有待分析的文档都上传到了服务器以后，就可以启动分析模块。聚类任务在后台进行，需要的时间比较长，该接口返回成功时仅表明分析任务已开始。

URL http://api.bosonnlp.com/cluster/analysis/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP GET 参数

参数名	取值范围	默认值	作用
alpha	(0, 1]	0.8	调节聚类最大 cluster 大小
beta	(0, 1)	0.45	调节聚类平均 cluster 大小

alpha 越大，最大产生的 cluster 会相应减小；beta 越大，形成 cluster 越不容易，平均 cluster 大小会减小。默认参数是在微博文本聚类上得到不错结果的一组经验参数，在其他应用（如新闻文本聚类）可能需要作相应的调整。

Tip: 聚类默认参数适用于在同一个关键词（例如“MH370”、“食品安全”等）对应的内容中发现细粒度子话题。对于其他类型的聚类需求，可以通过调节参数的方式调节聚类的粒度。

Java 调用代码示例

```
HttpResponse<String> clusterAnalysisResponse = Unirest
    .get("http://api.bosonnlp.com/cluster/analysis/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asString();
```

3.1.3 查看任务状态

一般文本聚类任务都用于处理大量的文本，需要比较长的处理时间。通过这个接口，可以查看聚类任务当前的状态信息。

URL http://api.bosonnlp.com/cluster/status/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 返回 Body 返回 JSON 格式的状态信息。可能的状态列表如下：

名字	说明
NOT FOUND	（在调用分析时）未找到任何数据
RECEIVED	成功接收到分析请求
RUNNING	数据分析正在进行中
ERROR	分析遇到错误退出
DONE	分析已完成

Java 调用代码示例

```
HttpResponse<String> clusterStatusResponse = Unirest
    .get("http://api.bosonnlp.com/cluster/status/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asString();
```

3.1.4 获取结果

一旦聚类引擎完成执行，结果将会被保存在服务器的数据库中。这个时候可以通过 HTTP GET 来取得聚类的结果数据。

URL http://api.bosonnlp.com/cluster/result/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 返回 Body 返回 JSON 格式的列表，包含以下字段：

字段	类型	说明
_id	与上传 _id 类型相同	该 cluster 最具代表性的文档
num	int	该 cluster 包含的文档数目
list	列表	所有属于该 cluster 的文档 _id

Java 调用代码示例

```
HttpResponse<JsonNode> clusterResultResponse = Unirest
    .get("http://api.bosonnlp.com/cluster/result/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asJson();
```

3.1.5 清除数据

清除 TASK_ID 对应的文本和聚类分析结果。

URL http://api.bosonnlp.com/cluster/clear/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 返回 Body

返回 **String** 格式的状态信息: Cleared %d documents.

详细的 Python SDK 文本聚类文档请看 这里 。

Java 调用代码示例

```
HttpResponse<String> clusterClearResponse = Unirest
    .get("http://api.bosonnlp.com/cluster/clear/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asString();
```

3.1.6 Python SDK 完整调用示例

```
# -*- encoding: utf-8 -*-
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')
docs = []


def print_cluster(idx, result):
    print '=' * 50
    print '第%d个聚类中共有%s份文档，如下:' % (idx + 1, result['num'])
    for ele in result['list']:
        print docs[ele]
    print '-' * 20
    print '本聚类的中心文档为:'
    print docs[result['_id']]


if __name__ == '__main__':
    with open('text_cluster.txt', 'r') as f:
        docs = [line for line in f if line]
```

```
    all_cluster = nlp.cluster(docs)
    sort_all_cluster = sorted(all_cluster, key=lambda cluster: cluster['num'], reverse=True)
    print sort_all_cluster
    for idx, cluster in enumerate(sort_all_cluster):
        print_cluster(idx, cluster)
```

详细的 Python SDK 文本聚类文档请看 这里 。

3.1.7 Java 完整调用示例

下面是 Java 的 Unirest 库 (Github) 调用 BosonNLP 的 Cluster API 的示例。

Tip: 完整的示例代码可以从 这里 下载。

```
package clusterApiExample;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;

import com.mashape.unirest.http.HttpResponse;
import com.mashape.unirest.http.JsonNode;
import com.mashape.unirest.http.Unirest;
import com.mashape.unirest.http.exceptions.UnirestException;

public class ClusterApiExample {
    // 请把 TASK_ID 换成您自己的 ID
    public static final String CLUSTER_PUSH = "http://api.bosonnlp.com/cluster/push/TASK_ID";
    public static final String CLUSTER_ANALYSIS = "http://api.bosonnlp.com/cluster/analysis/TAS
    public static final String CLUSTER_STATUS = "http://api.bosonnlp.com/cluster/status/TASK_I
    public static final String CLUSTER_RESULT = "http://api.bosonnlp.com/cluster/result/TASK_I
    public static final String CLUSTER_CLEAR = "http://api.bosonnlp.com/cluster/clear/TASK_ID"
    // 请记得把 YOUR_API_TOKEN 换成您申请的 Token
    public static final String API_TOKEN = "YOUR_API_TOKEN";
    public static String filePath = "clusterInfos.txt";
    public static boolean flag;
    public static void main(String[] args) throws JSONException,
                                                   UnirestException, IOException {
```

```
// 读取根目录下的 clusterInfos.txt 文件里的内容作为聚类上传的数据，请根据自身需求做更改
flag = readTxtFile(filePath);

// 聚类分析 API 调用
HttpResponse<String> clusterAnalysisResponse = Unirest
    .get(CLUSTER_ANALYSIS)
    .header("Accept", "application/json")
    .header("X-Token", API_TOKEN)
    .asString();
System.out.println(clusterAnalysisResponse.getCode());

// 调用查看状态的 API 来查看任务处理进度，这里每隔一段时间查询一次
while (flag) {
    // 聚类状态 API 调用
    HttpResponse<String> clusterStatusResponse = Unirest
        .get(CLUSTER_STATUS)
        .header("Accept", "application/json")
        .header("X-Token", API_TOKEN)
        .asString();
    JSONObject jsonObj = new JSONObject(clusterStatusResponse.getBody());
    String status = (String) jsonObj.get("status");

    try{
        Thread.sleep(2000);
    }catch(InterruptedException ie){
        ie.printStackTrace();
    }

    System.out.println(status);
    if ("DONE".equals(status)) {
        flag = false;
    }
    else
        System.out.println(" 请稍等，数据处理中...");
}

// 聚类结果 API 调用
HttpResponse<JsonNode> clusterResultResponse = Unirest
    .get(CLUSTER_RESULT)
    .header("Accept", "application/json")
    .header("X-Token", API_TOKEN)
    .asJson();
System.out.println(" 以下是返回的结果：");
System.out.println(clusterResultResponse.getBody());

// 聚类清除 API 调用
HttpResponse<String> clusterClearResponse = Unirest
```

```
            .get(CLUSTER_CLEAR)
            .header("Accept", "application/json")
            .header("X-Token", API_TOKEN)
            .asString();
        // 返回的结果是一段 JSON, num 指该 cluster 包含的文档数目, _id 指该 cluster 最具代表性的文档
        // list 指所有属于该 cluster 的文档 _id
        System.out.println(clusterClearResponse.getBody());
    }

    public static boolean readTxtFile(String filePath ) {
        try {
            String encoding = "utf-8";
            File file = new File(filePath);
            if (file.isFile() && file.exists()) { // 判断文件是否存在
                InputStreamReader reader = new InputStreamReader(new FileInputStream(file), enc
                BufferedReader bufferedReader = new BufferedReader(reader);
                String lineTxt = null;
                int i = 0;
                while ((lineTxt = bufferedReader.readLine()) != null) {
                    System.out.println(i);
                    JSONArray jsonArray = new JSONArray();
                    JSONObject jsonObject = new JSONObject();
                    jsonObject.put("_id", ++i);
                    jsonObject.put("text", lineTxt);
                    jsonArray.put(jsonObject);
                    String body = jsonArray.toString();
                    System.out.println(body);

                    // 聚类上传 API 调用
                    // 上传的数据时 JSON 格式的, 必须包含 _id(文档的编号, 可以采用整数或字符串) 和 text
                    HttpResponse<JsonNode> clusterPushResponse = Unirest.post(CLUSTER_PUSH)
                        .header("Accept", "application/json")
                        .header("Content-Type","application/json")
                        .header("X-Token", API_TOKEN)
                        .body(body).asJson();
                }
                reader.close();
                return true;
            }else{
                System.out.println(" 找不到指定的文件");
                return false;
            }
        } catch (Exception e) {
            System.out.println(" 读取文件内容出错");
            e.printStackTrace();
            return false;
        }
```

```
    }
}
```

构建

下载完后，把解压好的文件夹放到 Eclipse 的 Workspace 文件夹下，再新建一个名字一样的 Java Project。

运行

把 X-Token 改成您自己申请的 Token。把 TASK_ID 换成您自己的 ID。

`TASK_ID` 是任务的 ID，用于唯一识别该聚类任务，可由字母和数字组成。

3.1.8 应用场景示例

请看在 *MH370* 的微博中发现不同话题

3.2 典型意见引擎

用户意见在不同的网站，如微博评论，淘宝等电子商务网站评论里面都扮演量重要的角色。了解用户最典型的意见：

1. 从监测角度讲，可以快速了解事件传播中网民的核心论点，从而做出相应决策。
2. 从用户角度讲，能够快速掌握其他用户的意见，如其他用户对于某款产品的评价，帮助其选购。

例如，对某款香水的评论进行典型意见分析，会得到如下结果：味道很女人的 (256 类似)，瓶子的设计也很漂亮了 (237 类似)，留香时间很长 (156 类似)。

该引擎的使用需要以下三个步骤：

1. 上传需要聚类的文本数据；
2. 调用分析引擎（异步）；
3. 从服务器取得聚类的结果。

以下我们将按照上述的三个步骤来介绍引擎的调用。

3.2.1 上传数据

URL `http://api.bosonnlp.com/comments/push/TASK_ID`

`TASK_ID` 是任务的 ID，用于唯一识别该典型意见任务，可由字母和数字组成。

> **Warning:** 对于每次聚类任务应使用不同的 `TASK_ID` ，以免导致混淆。

HTTP Method `POST`

HTTP Header

Content-Type application/json

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 请求 Body JSON 格式的列表。每个元素为上传到聚类引擎的文档编号和内容组成的 JSON 对象，必须包含两个字段：

1. **_id** 文档的编号，可以采用整数或字符串
2. **text** 文档的内容，原始的文章信息

Note: 注意，在文本数量较大时，请分批上传数据。每批最多上传不超过 100 条文本。

Java 调用代码示例

```
HttpResponse<JsonNode> commentPushResponse = Unirest
    .post("http://api.bosonnlp.com/comments/push/TASK_ID")
    .header("Accept", "application/json")
    .header("Content-Type", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .body(YOUR_JSON_TEXT)
    .asJson();
```

3.2.2 调用分析

一旦所有待分析的文档都上传到了服务器以后，就可以启动分析模块。典型意见分析任务在后台进行，需要的时间比较长，该接口返回成功时仅表明分析任务已开始。

URL `http://api.bosonnlp.com/comments/analysis/TASK_ID`

HTTP Method `GET`

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP GET 参数

参数名	取值范围	默认值	作用
alpha	(0, 1]	0.8	调节聚类最大 cluster 大小
beta	(0, 1)	0.45	调节聚类平均 cluster 大小

alpha 越大，最大产生的 cluster 会相应减小；beta 越大，形成 cluster 越不容易，平均 cluster 大小会减小。默认参数是在微博典型意见上得到不错结果的一组经验参数，在其他应用（如新闻典型意见）可能需要作相应的调整。

Java 调用代码示例

```
HttpResponse<String> commentAnalysisResponse = Unirest
    .get("http://api.bosonnlp.com/comments/analysis/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asString();
```

3.2.3 查看任务状态

一般典型意见任务都用于处理大量的文本，需要比较长的处理时间。通过这个接口，可以查看典型意见任务当前的状态信息。

URL http://api.bosonnlp.com/comments/status/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 返回 Body 返回 JSON 格式的状态信息。可能的状态列表如下：

名字	说明
NOT FOUND	（在调用分析时）未找到任何数据
RECEIVED	成功接收到分析请求
RUNNING	数据分析正在进行中
ERROR	分析遇到错误退出
DONE	分析已完成

Java 调用代码示例

```
HttpResponse<String> commentStatusResponse = Unirest
    .get("http://api.bosonnlp.com/comments/status/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asString();
```

3.2.4 获取结果

一旦任务完成执行，结果将会被保存在服务器的数据库中。这个时候可以通过 HTTP GET 来取得典型意见分析的结果。

URL http://api.bosonnlp.com/comments/result/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 返回 Body 返回 JSON 格式的列表，包含以下字段：

字段	类型	说明
_id	与上传 _id 类型相同	该典型意见的标示
opinion	str	典型意见文本
num	int	该典型意见类似的意见个数
list	(str, int) 的列表	所有属于该典型意见的评论，其中 str 为意见，int 为意见的来源评论 ID

Note: 上述的返回结果中忽略了独立的评论，即每个典型意见至少包含 2 条评论。

Java 调用代码示例

```
HttpResponse<JsonNode> commentResultResponse = Unirest
    .get("http://api.bosonnlp.com/comments/result/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asJson();
```

3.2.5 清除数据

清除 TASK_ID 对应的文本和典型意见分析结果。

URL http://api.bosonnlp.com/comments/clear/TASK_ID

HTTP Method GET

HTTP Header

Accept application/json

X-Token YOUR_API_TOKEN （需要替换成您自己的 Token）

HTTP 返回 Body

返回 **String** 格式的状态信息: Cleared %d documents.

Java 调用代码示例

```
HttpResponse<String> commentClearResponse = Unirest
    .get("http://api.bosonnlp.com/comments/clear/TASK_ID")
    .header("Accept", "application/json")
    .header("X-Token", "YOUR_API_TOKEN")
    .asString();
```

3.2.6 Python SDK 完整调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')
docs = []


def print_comments(idx, comments):
    print '=' * 50
    print '第%d组典型意见是:' % (idx + 1)
    print comments['opinion']
    print '-' * 20
    print '共包含%s份文档，意见内容和原文 ID 如下:' % comments['num']
    for ele in comments['list']:
        print ele[0], ele[1]
```

```
if __name__ == '__main__':
    with open('text_comments.txt', 'r') as f:
        docs = [line for line in f if line]
    all_comments = nlp.comments(docs)
    sort_all_comments = sorted(all_comments, key=lambda comments: comments['num'], reverse=True
    for idx, comments in enumerate(sort_all_comments):
        print_comments(idx, comments)
```

详细的 Python SDK 典型意见文档请看 这里 。

3.2.7 Java 完整调用示例

下面是 Java 的 Unirest 库 (Github) 调用 Bosondata 的 Comment API 的示例。

Tip: 完整的示例代码可以从 这里 下载。

```
package commentApiExample;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;

import com.mashape.unirest.http.HttpResponse;
import com.mashape.unirest.http.JsonNode;
import com.mashape.unirest.http.Unirest;
import com.mashape.unirest.http.exceptions.UnirestException;

public class CommentApiExample {
    // 请把 TASK_ID 换成您自己的 ID
    public static final String COMMENT_PUSH = "http://api.bosonview.com/comments/push/TASK_ID"
    public static final String COMMENT_ANALYSIS = "http://api.bosonview.com/comments/analysis/T
    public static final String COMMENT_STATUS = "http://api.bosonview.com/comments/status/TASK_
    public static final String COMMENT_RESULT = "http://api.bosonview.com/comments/result/TASK_
    public static final String COMMENT_CLEAR = "http://api.bosonview.com/comments/clear/TASK_ID
    // 请记得把 YOUR_API_TOKEN 换成您的 Token
    public static final String API_TOKEN = "YOUR_API_TOKEN";
```

```
public static String filePath = "commentInfos.txt";
public static boolean flag;
public static void main(String[] args) throws JSONException,
                                               UnirestException, IOException {
    // 读取根目录下的 commentInfos.txt 文件里的内容作为典型意见上传的数据，请根据自身需求做更改
    flag = readTxtFile(filePath);

    // 典型意见分析 API 调用
    HttpResponse<String> commentAnalysisResponse = Unirest
        .get(COMMENT_ANALYSIS)
        .header("Accept", "application/json")
        .header("X-Token", API_TOKEN)
        .asString();
    System.out.println(commentAnalysisResponse.getCode());

    // 调用查看状态的 API 来查看任务处理进度，这里每隔一段时间查询一次
    while (flag){
        // 典型意见状态 API 调用
        HttpResponse<String> commentStatusResponse = Unirest
            .get(COMMENT_STATUS)
            .header("Accept", "application/json")
            .header("X-Token", API_TOKEN)
            .asString();
        JSONObject jsonObj = new JSONObject(commentStatusResponse.getBody());
        String status = (String) jsonObj.get("status");

        try{
            Thread.sleep(2000);
        }catch(InterruptedException ie){
            ie.printStackTrace();
        }

        System.out.println(status);
        if ("DONE".equals(status)) {
            flag = false;
        }
        else
            System.out.println(" 请稍等，数据处理中...");
    }

    // 典型意见结果 API 调用
    HttpResponse<JsonNode> commentResultResponse = Unirest
        .get(COMMENT_RESULT)
        .header("Accept", "application/json")
        .header("X-Token", API_TOKEN)
        .asJson();
    System.out.println(" 以下是返回的结果：");
```

```
        JsonNode result = commentResultResponse.getBody();
        String jsonData = result.toString();
        try {
            JSONArray jsonArray = new JSONArray(jsonData);
            String s = "";
            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObj = ((JSONObject) jsonArray.opt(i));
                int num = jsonObj.getInt("num");
                int id = jsonObj.getInt("_id");
                String op = jsonObj.getString("opinion");
                String list = jsonObj.getString("list");
                s += "num: " + num + ", _id: " + id + ", opinion: " + op + ", list: " + list +
            }
            // 返回的结果是一段 JSON，num 指该典型意见类似的意见个数，_id 指该典型意见的标示，opinion 指
            // list 指所有属于该典型意见的评论，其中 str 为意见，int 为意见的来源评论 ID
            System.out.println(s);
        } catch (JSONException ex) {
            // 异常处理代码
            ex.printStackTrace();
        }

        // 典型意见清除 API 调用
        HttpResponse<String> commentClearResponse = Unirest
            .get(COMMENT_CLEAR)
            .header("Accept", "application/json")
            .header("X-Token", API_TOKEN)
            .asString();
        System.out.println(commentClearResponse.getBody());
    }

    public static boolean readTxtFile(String filePath ) {
        try {
            String encoding = "utf-8";
            File file = new File(filePath);
            if (file.isFile() && file.exists()) { // 判断文件是否存在
                InputStreamReader reader = new InputStreamReader(new FileInputStream(file), enc
                BufferedReader bufferedReader = new BufferedReader(reader);
                String lineTxt = null;
                int i = 0;
                while ((lineTxt = bufferedReader.readLine()) != null) {
                    System.out.println(i);
                    JSONArray jsonArray = new JSONArray();
                    JSONObject jsonObject = new JSONObject();

                    jsonObject.put("_id", ++i);
                    jsonObject.put("text", lineTxt);
                    jsonArray.put(jsonObject);
```

```
                String body = jsonArray.toString();
                System.out.println(body);

                // 典型意见上传 API 调用
                // 上传的数据是 JSON 格式的，必须包含 _id(文档的编号，可以采用整数或字符串）和 text
                HttpResponse<JsonNode> commentPushResponse = Unirest
                    .post(COMMENT_PUSH)
                    .header("Accept", "application/json")
                    .header("Content-Type", "application/json")
                    .header("X-Token", API_TOKEN)
                    .body(body)
                    .asJson();
            }
            reader.close();
            return true;
        }else{
            System.out.println(" 找不到指定的文件");
            return false;
        }
    } catch (Exception e) {
        System.out.println(" 读取文件内容出错");
        e.printStackTrace();
        return false;
    }
  }
}
```

构建

下载完后，把解压好的文件夹放到 Eclipse 的 Workspace 文件夹下，再新建一个名字一样的 Java Project。

运行

把 X-Token 改成您自己申请的 Token。把 TASK_ID 换成您自己的 ID。

TASK_ID 是任务的 ID，用于唯一识别该聚类任务，可由字母和数字组成。

3.2.8 应用场景示例

请看汽车消费者典型意见

CHAPTER

FOUR

应用场景示例

4.1 找出最负面的消费者评价

4.1.1 介绍

我们随机在天猫上抽取 100 条关于某化妆品的评论作为分析数据，利用情感分析引擎来分析这 100 条数据，并且根据负面概率从大到小排序。负面概率结果为 0 到 1 之间的数值，通常负面概率大于 0.6 以上时，我们可以认定这条数据为负面。对于 0.4-0.6 之间的数据为模糊地带，由用户对这个区间的结果做判断，取一个相对的值，大于这个值的数据为负面。

Python 调用示例

请先下载测试数据：

text_sentiment.txt

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
import json
import requests


HEADERS = {'X-Token': 'YOUR_API_TOKEN'}
SENTIMENT_URL = 'http://api.bosonnlp.com/sentiment/analysis'


if __name__ == '__main__':
    print '读入数据...'
    with open('text_sentiment.txt', 'r') as f:
        docs = [line for line in f if line]
    print '正在上传数据...'
    for i in xrange(0, len(docs), 100):
        data = json.dumps(docs[i: i + 100])
```

```
        all_proba = requests.post(SENTIMENT_URL, headers=HEADERS, data=data.encode('utf-8')).js
        text_with_proba = zip(docs, all_proba)
        sort_text = sorted(text_with_proba, key=lambda element: element[1][1], reverse=True)
        for text in sort_text:
            print str(text[1]) + str(text[0])
```

查看完整的情感分析结果样例：sentiment_result.txt

4.2 在 MH370 的微博中发现不同话题

4.2.1 介绍

我们在新浪微博上搜索 # 马航 # 话题，随机抽取了 100 条微博来做聚类，虽然每条微博都在讲 # 马航 #，但是通过聚类引擎得出的结果可以看到 # 马航 # 话题中还是有不同的内容主体。

Python 调用示例

请先下载测试数据：text_cluster.txt

```
# -*- encoding: utf-8 -*-
import json
import requests
import sys
import time

CLUSTER_PUSH_URL = 'http://api.bosonnlp.com/cluster/push/'
CLUSTER_ANALYSIS_URL = 'http://api.bosonnlp.com/cluster/analysis/'
CLUSTER_RESULT_URL = 'http://api.bosonnlp.com/cluster/result/'
CLUSTER_CLEAR_URL = 'http://api.bosonnlp.com/cluster/clear/'
CLUSTER_STATUS_URL = 'http://api.bosonnlp.com/cluster/status/'
TASK_ID = 'Task%d' % time.time()

# 注意：在测试时请更换为您的 API token。
HEADERS = {'X-Token': 'YOUR_API_TOKEN'}


def raise_for_error(response):
    if not response.ok:
        response.reason = response.text
        response.raise_for_status()
```

```
def cluster_status():
    response = requests.get(CLUSTER_STATUS_URL + TASK_ID, headers=HEADERS)
    response.raise_for_error()
    return response.json()["status"]


def detail_results(idx, result):
    print '=' * 50
    print '第%d个聚类中共有%s份文档，如下:' % (idx + 1, result['num'])
    for ele in result['list']:
        print docs[ele]
    print '-' * 20
    print '本聚类的中心文档为:'
    print docs[result['_id']]


if __name__ == '__main__':
    print '任务 ID:', TASK_ID
    requests.models.Response.raise_for_error = raise_for_error

    print '读入数据...'
    with open('text_cluster.txt', 'r') as f:
        docs = [line for line in f if line]

    print '正在上传数据...'
    for i in xrange(0, len(docs), 100):
        data = json.dumps(
            [{'_id': idx, 'text': text} for idx, text in enumerate(docs[i:i+100])])
        requests.post(CLUSTER_PUSH_URL + TASK_ID, headers=HEADERS, data=data).raise_for_error()

    print '开始分析...'
    requests.get(CLUSTER_ANALYSIS_URL + TASK_ID, headers=HEADERS).raise_for_error()

    while True:
        status = cluster_status()
        if status == 'DONE':
            print '\n获取分析结果...'
            response = requests.get(CLUSTER_RESULT_URL + TASK_ID, headers=HEADERS)
            response.raise_for_error()
            all_cluster = response.json()

            print '一共生成了%d个聚类' % len(all_cluster)
            sort_all_cluster = sorted(all_cluster, key=lambda cluster: len(cluster['list']), re
            for idx, cluster in enumerate(sort_all_cluster):
                detail_results(idx, cluster)
```

```
            requests.get(CLUSTER_CLEAR_URL + TASK_ID, headers=HEADERS).raise_for_error()
            break
        elif status == 'NOT FOUND':
            print '找不到聚类任务。'
            break
        elif status == 'ERROR':
            print '任务失败，请稍后重试。'
            break
        else:
            sys.stdout.write('.')
            sys.stdout.flush()
            time.sleep(5)
```

Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')
docs = []


def print_cluster(idx, result):
    print '=' * 50
    print '第%d个聚类中共有%s份文档，如下:' % (idx + 1, result['num'])
    for ele in result['list']:
        print docs[ele]
    print '-' * 20
    print '本聚类的中心文档为:'
    print docs[result['_id']]


if __name__ == '__main__':
    with open('text_cluster.txt', 'r') as f:
        docs = [line for line in f if line]
    all_cluster = nlp.cluster(docs)
    sort_all_cluster = sorted(all_cluster, key=lambda cluster: cluster['num'], reverse=True)
    print sort_all_cluster
    for idx, cluster in enumerate(sort_all_cluster):
        print_cluster(idx, cluster)
```

详细的 Python SDK 文本聚类文档请看 这里 。

4.2.2 返回结果说明

```
>>>[{"_id": 46,                    //第 1 个聚类的中心文档 id
     "list": [3, 5, 44, 45, 46],  //该聚类包括的所有文档 id
     "num": 5                     //该聚类中的文档数量
     }...
```

查看完整的聚类结果样例：cluster_result.txt

4.3 汽车消费者典型意见

4.3.1 介绍

我们在某汽车论坛上对于某一型号汽车随机抽取了 100 条跟帖进行典型意见分析，通过典型意见引擎得出的结果可以看出该汽车在某些问题上面网友直观的评价，看法，以及对这些意见的聚类。

Python 调用示例

请先下载测试数据:

text_comments.txt

```
# -*- encoding: utf-8 -*-
import json
import requests
import sys
import time

COMMENTS_ANALYSIS_URL = 'http://api.bosonnlp.com/comments/analysis/'
COMMENTS_PUSH_URL = 'http://api.bosonnlp.com/comments/push/'
COMMENTS_RESULT_URL = 'http://api.bosonnlp.com/comments/result/'
COMMENTS_STATUS_URL = 'http://api.bosonnlp.com/comments/status/'
COMMENTS_CLEAR_URL = 'http://api.bosonnlp.com/comments/clear/'
TASK_ID = 'Task%d' % time.time()

# 注意：在测试时请更换为您的 API token。
HEADERS = {'X-Token': 'YOUR_API_TOKEN'}


def raise_for_error(response):
    if not response.ok:
```

```
        response.reason = response.text
        response.raise_for_status()


def comments_status():
    response = requests.get(COMMENTS_STATUS_URL + TASK_ID, headers=HEADERS)
    response.raise_for_error()
    return response.json()['status']


def detail_results(idx, comments):
    print '=' * 50
    print '第%d组典型意见是:' % (idx + 1)
    print comments['opinion']
    print '-' * 20
    print '共包含%s份文档，意见内容和原文 ID 如下:' % comments['num']
    for ele in comments['list']:
        print ele[0], ele[1]


if __name__ == '__main__':
    print '任务 ID:', TASK_ID
    requests.models.Response.raise_for_error = raise_for_error

    print '读入数据...'
    with open('text_comments.txt', 'r') as f:
        docs = [line for line in f if line]

    print('正在上传数据...')
    for i in xrange(0, len(docs), 100):
        data = json.dumps(
            [{'_id': i+idx, 'text': text} for idx, text in enumerate(docs[i:i+100])])
        requests.post(COMMENTS_PUSH_URL + TASK_ID, headers=HEADERS, data=data).raise_for_error

    print('开始分析...')
    requests.get(COMMENTS_ANALYSIS_URL + TASK_ID, headers=HEADERS)

    while True:
        status = comments_status()
        if status == 'DONE':
            response = requests.get(COMMENTS_RESULT_URL + TASK_ID, headers=HEADERS)
            response.raise_for_error()
            all_comments = response.json()

            print '一共生成了%d组典型意见' % len(all_comments)
            sort_all_comments = sorted(all_comments, key=lambda comments: len(comments['list'])
            for idx, comments in enumerate(sort_all_comments):
```

```
                detail_results(idx, comments)

            requests.get(COMMENTS_CLEAR_URL + TASK_ID, headers=HEADERS).raise_for_error()
            break
        elif status == 'NOT FOUND':
            print '找不到典型意见任务。'
            break
        elif status == 'ERROR':
            print '任务失败，请稍后重试。'
            print('开始分析...')
        else:
            sys.stdout.write('.')
            sys.stdout.flush()
            time.sleep(5)
```

Python SDK 调用示例

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals
from bosonnlp import BosonNLP

# 注意：在测试时请更换为您的 API token。
nlp = BosonNLP('YOUR_API_TOKEN')
docs = []


def print_comments(idx, comments):
    print '=' * 50
    print '第%d组典型意见是:' % (idx + 1)
    print comments['opinion']
    print '-' * 20
    print '共包含%s份文档，意见内容和原文 ID 如下:' % comments['num']
    for ele in comments['list']:
        print ele[0], ele[1]


if __name__ == '__main__':
    with open('text_comments.txt', 'r') as f:
        docs = [line for line in f if line]
    all_comments = nlp.comments(docs)
    sort_all_comments = sorted(all_comments, key=lambda comments: comments['num'], reverse=True
    for idx, comments in enumerate(sort_all_comments):
        print_comments(idx, comments)
```

详细的 Python SDK comments 文档请看 这里 。

4.3.2 返回结果说明

```
>>>[{"opinion": 起步离合太高,          //第 1 组典型意见的代表性意见
     "list": [[' 离合稍有点高',88]      //包含的意见内容和文档 ID
              [' 离合行程太高',110]
              [' 起步离合太高',215]
              [' 双离合效果不理想',229]]

     "num": 4                         //对应的文档数
     }...
```

查看完整的典型意见结果样例：`comments_result.txt`

CHAPTER

FIVE

其他

5.1 API 频率限制

5.1.1 介绍

BosonNLP API 根据不同的认证用户有不同的频率限制。该限制主要针对用户上传的篇数以及 API 调用次数。

每次 API 请求返回的 HTTP header 里包括了该 API 限制的详细信息，详情看这里。也可以通过 rate limit 接口来查询用户所有 API 限制的详细信息。

篇数 一天上传文档的数量，东八区每天 0 点重置

次数 每 15 分钟调用接口次数，一次 http 请求为调用一次该接口

Note: 每 15 分钟重置时间为每小时的 0 分，15 分，30 分，45 分。

5.1.2 查询接口介绍

本接口用来查询用户使用 BosonNLP API 频率限制的详细信息。

URL `http://api.bosonnlp.com/application/rate_limit_status.json`

HTTP Method `GET`

HTTP Header

`X-Token` `YOUR_API_TOKEN` （需要替换成您自己的 Token）

HTTP 返回 Body 返回 JSON 格式的状态信息

5.1.3 CURL 调用示例

```
$ curl -H "X-Token: YOUR_API_TOKEN" http://api.bosonnlp.com/application/rate_limit_status.json
```

5.1.4 Python 调用示例

```
# -*- encoding: utf-8 -*-
import requests

# 注意：在测试时请更换为您的 API token。
HEADERS = {'X-Token': 'YOUR_API_TOKEN'}
RATE_LIMIT_URL = 'http://api.bosonnlp.com/application/rate_limit_status.json'

result = requests.get(RATE_LIMIT_URL, headers=HEADERS).json()
```

5.1.5 返回结果说明

```
{
"limits": {
    "classify": {                               //API 名称
        "count-limit-limit": 3000000,           //今天可上传文档的篇数总量
        "count-limit-remaining": 3000000,       //今天剩余可上传文档的篇数
        "count-limit-reset": 1399910400,        //今天重置上传文档篇数限制的剩余时间
        "rate-limit-limit": 50000,              //15 分钟内可调用接口总次数
        "rate-limit-remaining": 50000,          //15 分钟内剩余可调用接口次数
        "rate-limit-reset": 1399868100          //15 分钟内调用次数限制的重置时间
    }...
```

字段	类型	说明
count-limit-limit	int	今天可上传文档的篇数总量，一天的开始从东八区的 0 点开始计算
count-limit-remaining	int	今天剩余可上传文档的篇数
count-limit-reset	unix timestamp	今天重置上传文档篇数限制的剩余时间
rate-limit-limit	int	15 分钟内可调用接口总次数
rate-limit-remaining	int	15 分钟内剩余可调用接口次数
rate-limit-reset	unix timestamp	重置调用次数限制的剩余时间

5.1.6 API 调用失败说明

用户遇调用 API 次数受限制时会遇到如下 HTTP 返回信息:

```
{"status":429,"message":"count/rate limit exceeded"}
```

请确认 X-Count-Limit-Remaining 和 X-Rate-Limit-Remaining 还有剩余调用量，否则要等待相应的重置时间才可以继续调用 API。

X-Count-Limit-Remaining 为 0 时，东八区 0 点重置。

X-Rate-Limit-Remaining 为 0 时，参考前文 API 调用次数重置时间。

5.2 HTTP 状态码列表

BosonNLP HTTP API 的每个请求均返回合适的 HTTP 状态码 。

200 调用成功

400 HTTP 请求不满足要求。

403 Token 验证失败，具体的失败原因见返回的 JSON。

413 上传文档超过 100 篇上限。

```
{"status":413,"message":"maximum size of 100 exceeded"}
```

或者 HTTP 请求超过 2M 的大小限制。

```
HTTP/1.1 413 Request Entity Too Large
Date: Thu, 07 Aug 2014 09:17:30 GMT
Content-Type: text/html
Content-Length: 192
Connection: close
Server: nginx/1.5.11
X-Request-Id: 114e70e3-7d28-4651-83df-4e9da622062f

<html>
  <head><title>413 Request Entity Too Large</title></head>
  <body bgcolor="white">
    <center><h1>413 Request Entity Too Large</h1></center>
    <hr><center>nginx</center>
  </body>
</html>
```

429 超出每 15 分钟或者每天的调用额度。

500 服务故障。

502 服务故障或者在升级中。

503 服务正常，但是您的请求并发数或者连接数超过了我们系统设定的阈值。

5.2.1 错误信息说明

当调用 API 发生错误时，将返回 JSON 格式的错误信息。

```
{"status":403,"message":"no token header"}
```