Apache openNLP 简介

作者：felayman
原文链接：https://ld246.com/article/1519818629879
来源网站：链滴

简介

Apache OpenNLP库是一种基于机器学习的工具包，用于处理自然语言文本。它支持最常见的NLP务，如标记化，句子分割，词性标记，命名实体提取，分块，解析和参考解析。 通常需要这些任务来建更高级的文本处理服务。 OpenNLP还包括基于最大熵和感知器的机器学习。

OpenNLP项目的目标是为上述任务创建一个成熟的工具包。 另一个目标是为各种语言提供大量预构的模型，以及这些模型来自的注释文本资源。

Apache OpenNLP库包含几个组件，使得能够构建一个完整的自然语言处理管道。 这些组件包括：子检测器，分词器，名称查找器，文档分类器，词性标记器，chunker，解析器，参数解析。组件包能够执行各自然语言处理任务，训练模型以及通常还用于评估模型的部分。 每个这些设施都可以通过应用程序接口（API）访问。 此外，提供命令行界面（CLI）以方便实验和训练。

我们可以利用Apache OpenNLP的NLP任务,比如分词、断句、词性标注、命名实体抽取、组块分析解析和指代消解。通常这些任务用来提供高级文本处理服务。

主要功能

- 句法检测器： Sentence Detector
- 分词器： Tokenizer
- 名字查找器： Name Finder
- 文档分类器： Document Categorizer
- 词性标注器： Part-of-Speech Tagger
- 组块分析器： Chunker
- 组块分析器： Chunker
- 指代消解： Coreference Resolution

官方网站地址:http://opennlp.apache.org/

使用流程

- 1.训练模型，当然也可以跳过该步骤，而直接使用官网提供的一些已经训练好的模型
- 2.加载模型，不同任务需要加载不同的模型，通常都是通过构造一个InputStream来加载训练好的型;
- 3.构造预测器，通过加载进来的训练好的模型构造预测器;
- 4.利用预测器进行NLP相关任务

入门实例

引入maven

```
<dependency>
  <groupId>org.apache.opennlp</groupId>
  <artifactId>opennlp-tools</artifactId>
  <version>1.8.</version>
```

原文链接：Apache openNLP 简介

<dependency>

下载opennlp的训练模型(models)

下载地址:opennlp-models

说明

虽然上述models的版本都是1.5,同样兼容opennlp 1.8 版本

引入模型

这里我将上述下载的训练模型放入到maven项目的resources目录下,我下载的是en-token.bin,就是用对英文进行token化的模型.

代码如下:

```
@Test
    public void test(){
        try (InputStream modelIn = new FileInputStream("/Users/admin/work/vcg-test-parent/v
g-kotlin-test/src/main/resources/de-token.bin")) {
            TokenizerModel tokenizerModel = new TokenizerModel(modelIn);
            TokenizerME tokenizerME = new TokenizerME(tokenizerModel);
            Arrays.stream(tokenizerME.tokenize("to be or not to be."))
                    .forEach(System.out::println);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

结果如下:

```
to
be
or
not
to
be
.
```

我们可以看到对原内容没有进行任何修改,只是以空格为停顿词进行切割.

术语解析

- Language Detector 语言嗅探器
- Sentence Detector 语法嗅探器
- Tokenizer 分词器
- Name Finder 名字查找器
- Document Categorizer 文档分类器

- Part-of-Speech Tagger 词性标注器
- Lemmatizer 词形还原器
- Chunker 组块分析器
- Parser 解析器

工具使用

语言嗅探器(Language Detector)

OpenNLP语言嗅探器根据模型功能将ISO-639-3语言中的文档分类。一个模型可以用最大熵，感知或朴素贝叶斯算法来训练。默认情况下利用n-grams的1, 2 和 3距离(空间编辑距离)，来抽取大小为1 2和3的分词。通过扩展语言detectorfactory，可以定制n-gram大小、规范化和上下文生成器。

标准化因子(Normalizer)

OpenNLP默认的标准化因子有:

- EmojiCharSequenceNormalizer 将emojis表情符用空格替换
- UrlCharSequenceNormalizer 将E-Mail地址以及url用空格替换
- TwitterCharSequenceNormalizer 将标签和Twitter的用户名用空格替换
- NumberCharSequenceNormalizer 将数字串用空格替换
- ShrinkCharSequenceNormalizer 将多个重复(大于三个)的字符压缩成2个

下面是测试的API

```
@Test
    public void langdetect() {
        try (InputStream modelIn = new FileInputStream("/Users/admin/work/vcg-test-parent/v
g-kotlin-test/src/main/resources/langdetect-183.bin")) {
            LanguageDetectorModel m = new LanguageDetectorModel(modelIn);
            LanguageDetector languageDetectorME = new LanguageDetectorME(m);
            Language bestLanguage = languageDetectorME.predictLanguage("测试语言嗅探器");
            System.out.println("Best language: " + bestLanguage.getLang());
            System.out.println("Best language confidence: " + bestLanguage.getConfidence());
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

结果如下:

```
Best language: cmn
Best language confidence: 0.010829125435641998
```

OpenNLP语言嗅探器会根据模型来计算出一个置信度,来表明该语言最有可能是哪种语言.内置的语言析器会以组为单位来分析出输入的文本内容的语言分布,置信度的大小即表示语言的可信度.

通过结果我们发现,我们输入的全是中文,为什么出来的置信度只有这么低,我们接下来就做一件事情,重训练模型,来增大该置信度

句子嗅探器(Sentence Detection)

```
@Test
    public void sentence() {
        try (InputStream modelIn = new FileInputStream("/Users/admin/work/vcg-test-parent/vg-kotlin-test/src/main/resources/en-sent.bin")) {
            SentenceModel model =new SentenceModel(modelIn);
            SentenceDetectorME sdetector =new SentenceDetectorME(model);
            Arrays.stream(sdetector.sentDetect("Hi. How are you? This is Mike. Who are you?"))
                    .forEach(System.out::println);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

结果如下:

```
Hi. How are you?
This is Mike.
Who are you?
```

分词器(Tokenization)

```
@Test
    public void tokenization() {
        try (InputStream modelIn = new FileInputStream("/Users/admin/work/vcg-test-parent/vg-kotlin-test/src/main/resources/en-token.bin")) {
            TokenizerModel model = new TokenizerModel(modelIn);
            Tokenizer tokenizer = new TokenizerME(model);
            Arrays.stream(tokenizer.tokenize("An input sample sentence."))
                    .forEach(System.out::println);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

结果如下:

```
An
input
sample
sentence
.
```

文档分类器(Document Categorizer)

参考

- OpenNLP：驾驭文本，分词那些事
-

原文链接：Apache openNLP 简介