《大数据开发技术（Hadoop）》

实训报告

主题：互联网新闻自动归类项目实战

组长：欧阳乐桦

成员：苏俊健，梁珈睿

完成日期：2025/1/4

目 录

[第1部分 项目实现技术 1](#_Toc27978)

[第2部分 项目实现思路介绍 1](#_Toc20657)

[第3部分 项目实现流程介绍 1](#_Toc22979)

[第4部分 项目分析结果说明 1](#_Toc13545)

[第5部分 项目代码 2](#_Toc1485)

# 项目实现技术

Java

WebMagic

# 项目实现思路

1.数据采集

2.数据清洗

3.数据分析（可视化）

# 项目实现流程介绍

3.1**数据采集**

数据收集



以同样方法检查其他新闻网址，找到这些新闻详细页网址的规律，发现新闻标题的格式为：/qqdetail/5,即网址都有公共部分/qqdetail/，不同的地方是后面的数字。找到这些网址的规律，是进行爬虫实验的前提工作。

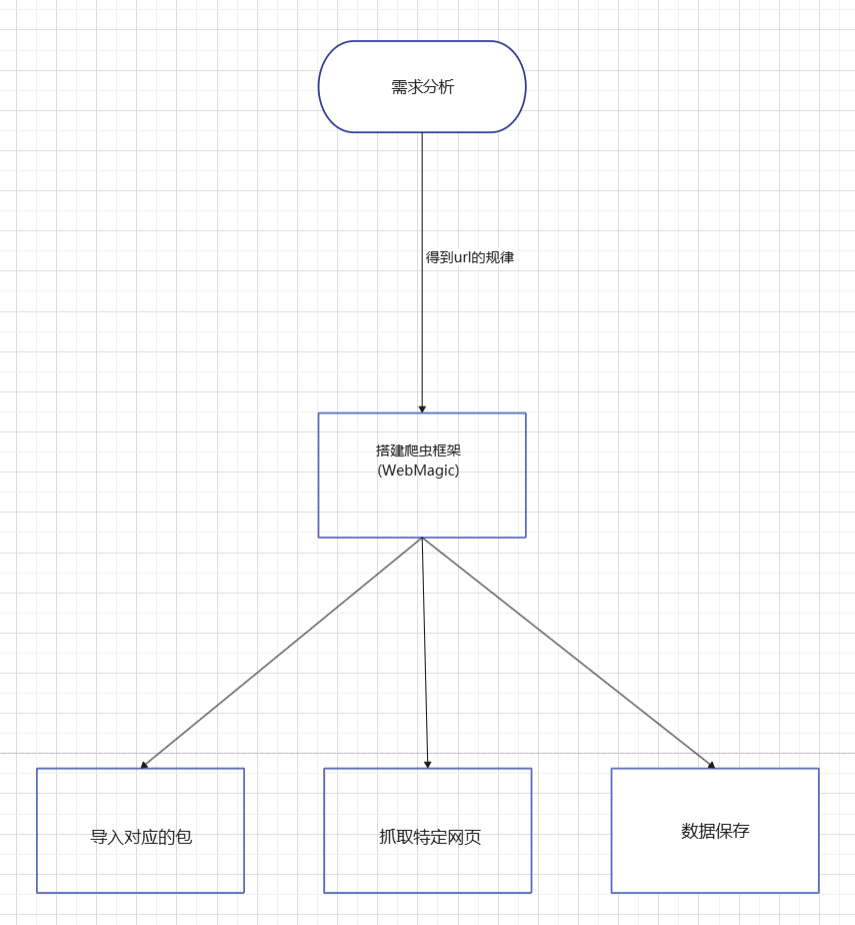
数据抓取

因此在数据抓取我们将体育类新闻的详细页通过addTargetRequests抓取网页，详细页中不同网页链接的不同之处是数字部分，我们可以用正则表达式匹配。

page.addTargetRequests(page.getHtml().links().regex("(/qqdetail/\\d+)").all());

数据存储

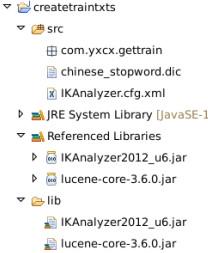
在main函数中，通过Spider来进行爬虫启动，将数据存储在文件中。



**3.2数据分析处理**

3.2.1 训练模型

项目搭建，项目结构如下图所示：



3.2.2 网页预处理

（1）网页包含大量的结构化标签，比如<head>，<title>等，我们要保证标签不冗杂，因此需要处理。

（2）网页中包含着大量噪音，包括各种广告、导航、注释以及版权申明等一 些和内容无关的东西，在分类之前需要去除这些噪音，否则这些噪音会 影响分类性能。

（3）超链接网页中存在大量的超链接。超链接将互联网上的网页连成了一张巨大的 网络，网页上的超链接代表着这个网页到另一个网页的路径，通过超链接我们可以获得网页的一些特征，比如索引页就存在大量的超链接。因此我们需要去除这一部分

3.2.3 中文分词

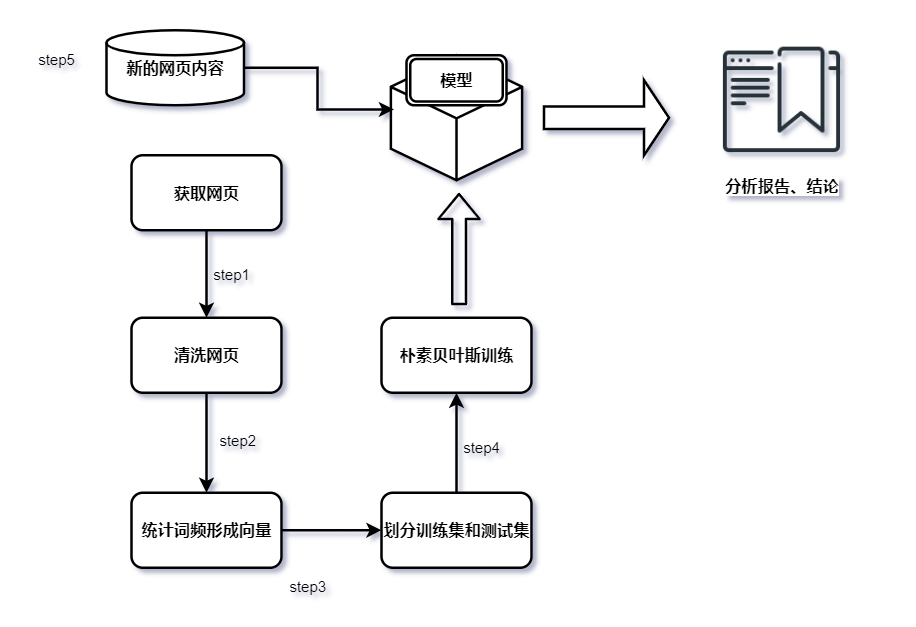
中文分词指的是将一个汉字序列切分成一个一个单独的词。分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。

3.2.4 页面特征提取

网页分类的质量很大程度上取决于特征提取的好坏，网页本身具有复杂无规律的很多特征，而我们需要对这些特征进行提取筛选，选取那些具有区分度的特征，特征的选取主要是通过对网页的预处理提取出网页的文本信息和结构信息，然后利用一些成熟的特征筛选方法进行筛选，包括特征频率(TF)、文档频率 (DF)、信息增益(IG)、互信息(MI)、卡方拟和检验(CHI)以及期望交叉熵(ECE)

**3.3基于朴素贝叶斯算法的网页分类**

流程图如下：



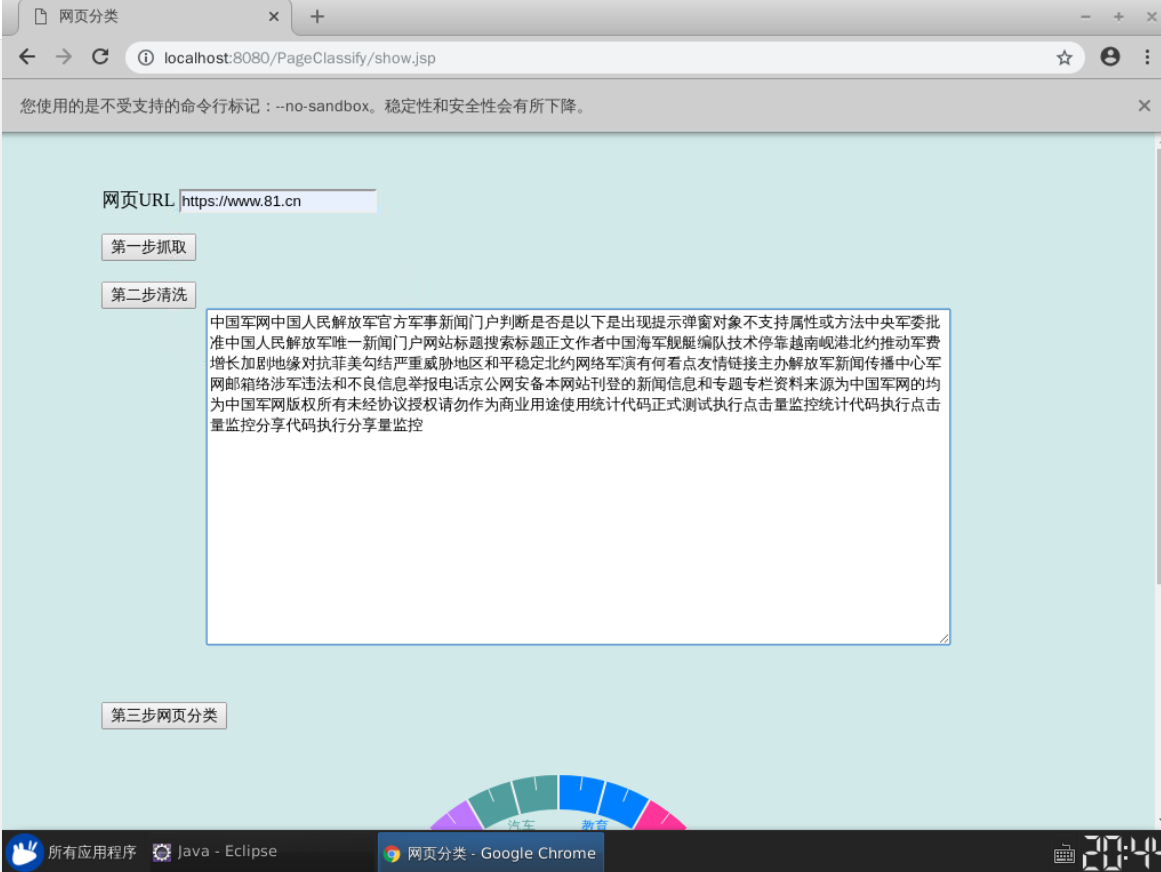
关于模型的一些说明：

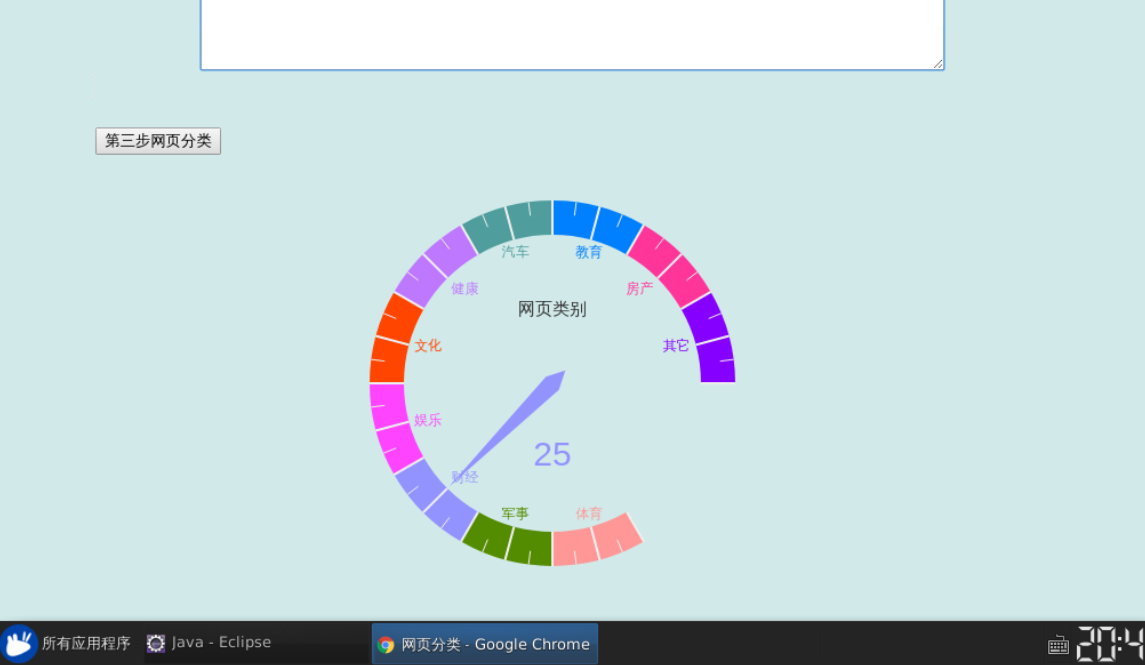
1. 平台上给出的java代码，使用的是词-文档（页面）共现算法构造稀疏向量。只统计在词库中且在一个页面中出现**频数大于3**的词的个数。
2. 不考虑词于词之间的位置，词所在的语境以及词与词之间的联系等等，各个词之间假设是**独立**的。
3. 通过**极大似然估计**来学习朴素贝叶斯模型，并运用**拉普拉斯平滑**的方法避免极端边界情况造成的巨大影响



# 项目分析结果说明

模型分类结果准确率不高，例如下列输入<https://www.81.cn>（中国军网），本应归类为军事类，可模型结果却是财经类，说明模型整体较为粗糙，分类效果差强人意。





**改进方案**：

1. 在构造词向量时，应该充分考虑到词与词之间的联系，词于上下文的联系等等，有利于整体把握文本情感色彩，语境。
2. 增大训练集规模，提升模型拟合能力。
3. 使用Word2Vec，Doc2Vec等算法构造稠密词向量，充分考虑到词与词间的关联，利用基于深度学习的方法建模。

# 项目代码

Word2Vec代码示例

import org.deeplearning4j.models.embeddings.loader.WordVectorSerializer;  
 import org.deeplearning4j.models.word2vec.Word2Vec;  
 import org.deeplearning4j.text.sentenceiterator.BasicLineIterator;  
 import org.deeplearning4j.text.tokenization.tokenizer.preprocessor.CommonPreprocessor;  
 import org.deeplearning4j.text.tokenization.tokenizerfactory.DefaultTokenizerFactory;  
 import org.deeplearning4j.text.tokenization.tokenizerfactory.TokenizerFactory;  
 import java.io.File;  
  
  
public class DocumentToDenseVectors {  
  
 public static Word2Vec generateWordVectors(String inputFilePath) {  
 // 使用 BasicLineIterator 从文件中逐行读取文本  
 BasicLineIterator iterator = new BasicLineIterator(inputFilePath);  
 // 创建一个默认的分词器工厂，并使用 CommonPreprocessor 对分词进行预处理  
 TokenizerFactory tokenizerFactory = new DefaultTokenizerFactory();  
 tokenizerFactory.setTokenPreProcessor(new CommonPreprocessor());  
  
 // 构建 Word2Vec 模型  
 Word2Vec word2Vec = new Word2Vec.Builder()  
 .minWordFrequency(1) // 最小词频，低于此频率的词将被忽略  
 .layerSize(100) // 词向量的维度  
 .seed(42) // 随机种子，用于结果的可重复性  
 .windowSize(5) // 上下文窗口大小  
 .iterate(iterator) // 使用迭代器进行训练  
 .tokenizerFactory(tokenizerFactory) // 使用定义的分词器工厂  
 .build();  
  
 // 训练 Word2Vec 模型  
 word2Vec.fit();  
  
 return word2Vec;  
 }  
  
 public static void main(String[] args) {  
 String inputFilePath = "path/to/your/document.txt";  
 Word2Vec word2VecModel = generateWordVectors(inputFilePath);  
  
 // 获取单词的词向量  
 double[] vector = word2VecModel.getWordVector("example");  
 System.out.println("Vector for 'example': " + java.util.Arrays.toString(vector));  
  
 // 保存模型，以便后续使用  
 WordVectorSerializer.writeWord2VecModel(word2VecModel, new File("word2vec.model"));  
 }  
}

Doc2Vec代码示例

import org.deeplearning4j.models.doc2vec.Doc2Vec;  
 import org.deeplearning4j.models.doc2vec.LabelAwareIterator;  
 import org.deeplearning4j.models.doc2vec.TaggedDocument;  
 import org.deeplearning4j.text.sentenceiterator.BasicLineIterator;  
 import org.deeplearning4j.text.tokenization.tokenizer.preprocessor.CommonPreprocessor;  
 import org.deeplearning4j.text.tokenization.tokenizerfactory.DefaultTokenizerFactory;  
 import java.io.File;  
 import java.util.ArrayList;  
 import java.util.List;  
  
  
public class Doc2VecExample {  
 public static void main(String[] args) {  
 String filePath = "path/to/your/document.txt";  
 BasicLineIterator iterator = new BasicLineIterator(filePath);  
 DefaultTokenizerFactory tokenizerFactory = new DefaultTokenizerFactory();  
 tokenizerFactory.setTokenPreProcessor(new CommonPreprocessor());  
  
  
 List<TaggedDocument> documents = new ArrayList<>();  
 int i = 0;  
 while (iterator.hasNext()) {  
 String line = iterator.next();  
 documents.add(new TaggedDocument(tokenizerFactory.create(line).getTokens(), new int[]{i++}));  
 }  
  
  
 Doc2Vec doc2Vec = new Doc2Vec.Builder()  
 .minWordFrequency(1)  
 .layerSize(100)  
 .seed(42)  
 .windowSize(5)  
 .iterations(10)  
 .epochs(1)  
 .build();  
  
  
 doc2Vec.fit(new LabelAwareIterator() {  
 @Override  
 public boolean hasNextDocument() {  
 return iterator.hasNext();  
 }  
  
  
 @Override  
 public TaggedDocument nextDocument() {  
 return documents.get(iterator.next());  
 }  
  
  
 @Override  
 public void reset() {  
 iterator.reset();  
 }  
  
  
 @Override  
 public int totalNumDocs() {  
 return documents.size();  
 }  
  
  
 @Override  
 public int numDocs() {  
 return documents.size();  
 }  
  
  
 @Override  
 public List<String> labelsSource() {  
 return null;  
 }  
 });  
  
  
 // 获取文档的向量表示  
 double[] docVector = doc2Vec.inferVector(tokenizerFactory.create("This is an example document").getTokens());  
 System.out.println(java.util.Arrays.toString(docVector));  
 }  
}

参考目录如下：