实验报告 – Spark/Hadoop for Natural language processing

周瑶 15212158

实验介绍

随着互联网和计算机硬件的迅速发展，大数据随之产生。互联网上每天都会产生很多种类的大量的数据信息，常见的有图片，文本，视频等… 而分析处理这些数据，从而得到有价值的信息，已成为学术界和工业界的热点。在人工智能领域，计算机视觉和自然语言处理(计算语言学)是针对互联网上生产的自然信息的两种主要的研究方向。而针对海量的文本数据，传统的分析方法显得效率不足，自从2004年Google开源Hadoop以来，它迅速成为开源社区的讨论热点，并且成为大数据处理方面的首先开源工具。由于Hadoop的MapReduce的局限性，University of Calfornia, Berkeley的AMP Lab开源了一个高性能可扩展的分布式计算框架-Spark, Spark因为其高效性和灵活性，迅速成为大数据处理的新宠。在国内外成为仅次于Hadoop的分布式并行计算框架。

深度学习(Deep Learning)在经历过20世纪90年代的衰落后，在近几年得到迅速复兴(2009-present)。深度学习在各个方法表现杰出，事实证明，随着硬件水平的提高，当计算资源不再是研究的瓶颈时，深度学习能够拟合现实世界中的大部分模型。在自然语言处理中，命令实体识别和情感分析都是较高阶的任务。文本的实体识别能找到语言中提及的实体，比如商品产品，日期，事件等，而情感分析则是对自然语言的表达进行分析，分析出语言中的情感极性。这些研究都对现实生活有着巨大帮助。

在以上背景下，我完成了此次项目，将大规模分布式计算框架与自然语言中的深度学习相结合，对电商网站(亚马逊)的评论进行命令实体识别与情感极性分析。亚马逊评论的数据量巨大，分析产品评论情感极性非常具有挑战，于是我将Spark/Hadoop用于处理大规模数据，将深度学习用于情感分析。实验证明，这种方式极大增加了效率，并且取得了不错的成果。

在本实验中用到的算法，如下表:

算法	算法细节	复杂度
分词	正则表达式，采用Stanford CoreNLP的tokenize和ssplit	较低
实体识别	统计学方法，随机条件场(CRF), Stanford CoreNLP	中等
情感分析	深度学习，Recursive Neural Network, Stanford CoreNLP	较高
MapReduce	Map, FlatMap, Reduce, Foreach	较高

如何运行

由于依赖包太多，文件太大，所以我采用Maven作为包管理工具。整个项目的依赖在程序目录的pom.xml。
运行指令如下:

```
cd ../haruka
mvn package
cd target/
chmod 777 run
./run spark -i [input_file]
```

实验设施

- 数据集: Amazon products reviews 50G+;
- 计算资源: 6台Unix/Linux服务器的集群，三台I5四核16GB内存配SSD，三台I5四核8GB内存配普通硬盘。所有节点配有Java8，Spark1.5.2，Hadoop2.6。
- 存储资源: Redis, ElasticSearch，MongoDB以及MySQL。
- 由于集群都是在一个局域网内，并且通过一个小型交换机有线连接起来，可以保证高速无障碍通信。

在本次实验中，我用6台集群先跑了10G的数据，耗时近12个小时，因为我项目比较复杂，计算复杂的较高，我认为花费这么多时间是合理的。之后我采用全部50G数据，当跑完两天(20+小时)之后，考虑到这个任务耗费太多计算资源，影响其他工作，所以就中止了。我估算整个项目跑完大概需要40个小时左右。

项目细节

在本项目中，我采用了许多开源工具包，整体工程量也较大，接下来我会详解介绍在本项目的结构细节。

结构

我把项目命名为haruka，并将其分成了7个Package(见附录1)，具体介绍如下:

> - `cli`，命令行工具包，主要用于通过脚本语言执行java程序，在本项目中我使用Ruby作为调用的脚本语言。
> - `core`， core包主要存放一些配置变量，包括sparkconfig和其他信息。
> - `instance` instance主要是亚马逊评论类和分析完的评论类。
> - `nlp` nlp包括自然语言处理的工具，tokenize，实体识别和情感分析。
> - `nosql` nosql是redis和elasticsearch两个NoSQL数据库的客户端接口。
> - `run` 运行Spark任务的接口。
> - `utils` 工具类，包括各种需要的工具接口。

流程

整个项目run脚本来执行，具体的运行流程如附录2所示。 在run脚中指定了通过命令行执行Java程序的的命令，当启动任务的时候，只需要告诉脚本输入文件的路径，然后脚本执行 `cli.SparkCli` 来调用 `TaskRunner` ，在 `TaskRunner` 中程序将读入的输入文件路径作为参数，将输入文件中的文本分成若干块在并行执行，对于文件中的每一行，将json格式解析成为 `AmazonReview` 的实体，再将解析完的实体进行逐个分析，抽取中每段文本中的命名实体、情感分析极性。命名实体抽取和情感分由在nlp包中的 `EntityRecognize` 和 `Sentiment` 完成。最后将分析完的数据存入 `redis` 中， `redis` 和 `elasticsearch` 的java接口在 `nosql` 包中简单实现。

实验结果与分析

实验结果数据包含两部分的数据，一部分为原始数据中的部分，另一部分为经过分析抽取后的数据。我将分析后的数据也存为Json格式，它的结构如下所示:

```
{
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "I bought this for my husband who plays the piano. Having a wonderful time"
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "reviewTime": "09 13, 2009",
  "entity": {"pinao": "instrument"},
  "sentiment": 3
}
```

然后将json格式的数据转化成字符串，存入redis的set结构中进行自动去重。

实验总结

在本次实验中，我采用了Spark+hadoop的分布式并行计算框架，Stanford CoreNLP的自然语言处理工具包，redis+elasticsearch的分布式NoSQL数据库来完成整个项目。将电商网站的原始产品评论，经过分析得到评论中的评价实体和评论的情感极性，达到了实际的工业级应用的水准。

在整个项目中，我采用Java作为主要的工程语言，配合ruby脚本语言，完成了整个编码部分。在集群的搭建方面，采用了多台Linux/Unix作为集群的服务器，搭建了小型的集群，具有小规模并行计算的能力。通过整个项目，我对spark和hadoop以及其他NoSQL又有了更加深入的了解。

参考文献

[Spark - Lightning-fast cluster computing.]

[Hadoop 分布式并行计算框架]

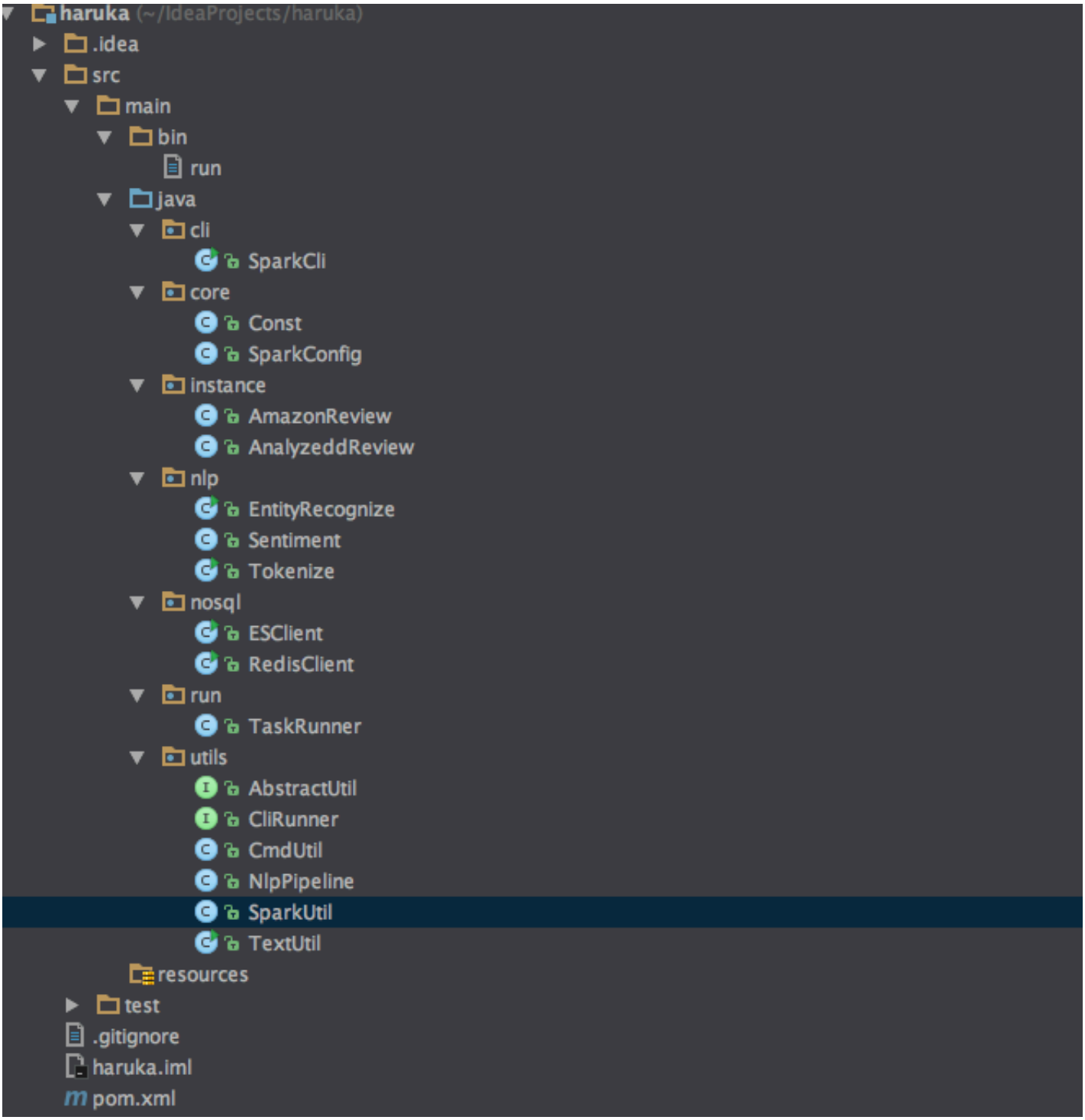
[ElasticSearch Java Clien API]

[Redis 参考文档]

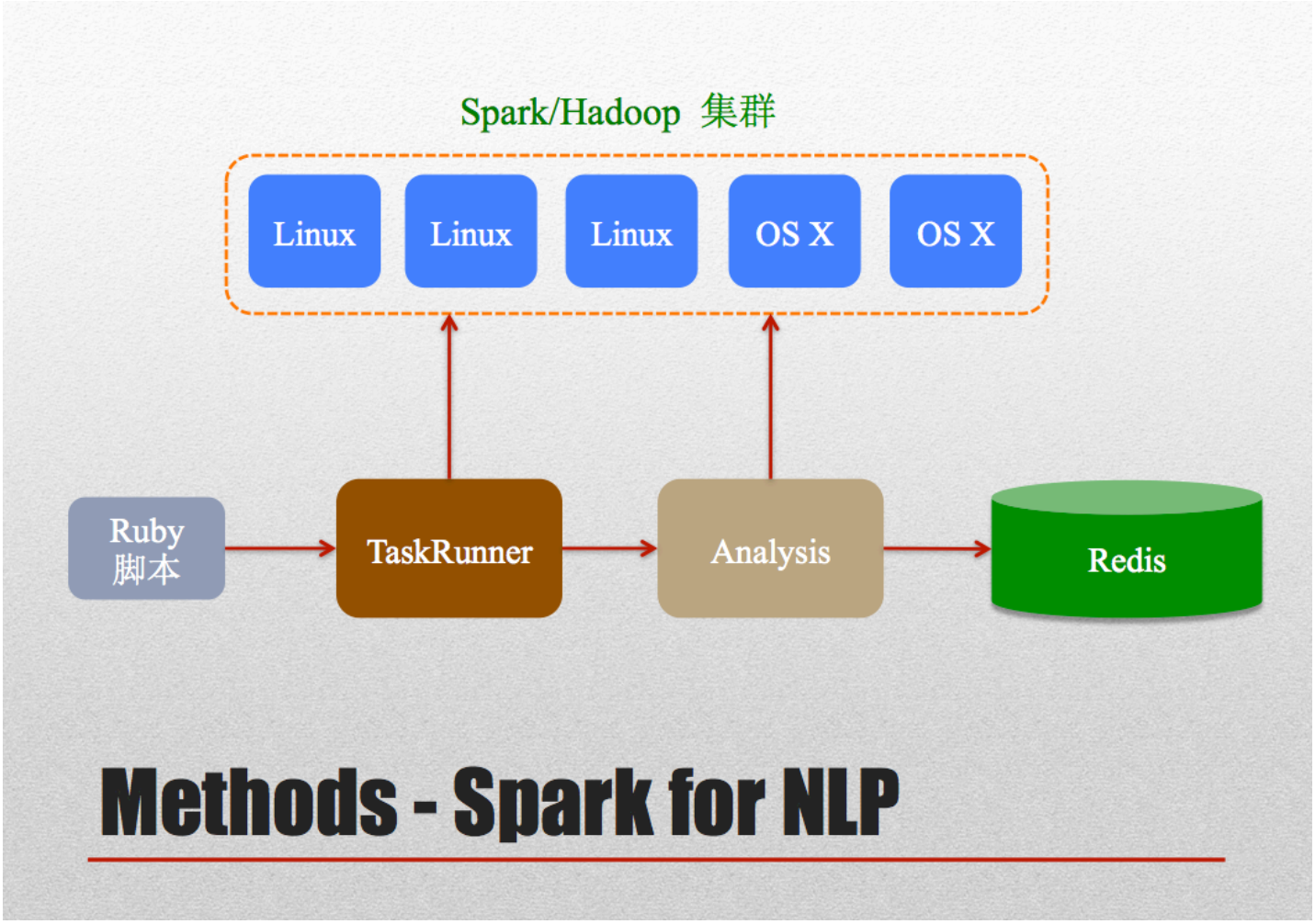
[Stanford CoreNLP toolkits]

[Recursive Neural Network for Sentiment classification]

附录1: 项目结构图



附录2: 项目流程图



附录3:Map和foreach的java实现:

```
    public static class ParseFunction implements Function<String, AmazonReview> {
        @Override
        public AmazonReview call(String s) throws Exception {
            return gson.fromJson(s, type);
        }
    }

    public static class AnalyzeVoidFunction implements VoidFunction<AmazonReview> {
        String review, user, summary, reviewTime, helpful;
        double overall;
        AnalyzeddReview anreview = new AnalyzeddReview();

        @Override
        public void call(AmazonReview amazonReview) throws Exception {
            review = amazonReview.getReviewText();
            user = amazonReview.getReviewerName();
            helpful = Arrays.toString(amazonReview.getHelpful());
            summary = amazonReview.getSummary();
            reviewTime = amazonReview.getReviewTime();
            overall = amazonReview.getOverall();
            int sentiment = Sentiment.getSentiment(review);
            Map<String, String> entity = EntityRecognize.getNamedEntity(review);
            anreview.setEntity(entity);
            anreview.setHelpful(helpful);
            anreview.setOverall(overall);
            anreview.setReviewerName(user);
            anreview.setReviewTime(reviewTime);
            anreview.setSentiment(sentiment);
            anreview.setSummary(summary);
            anreview.setReviewText(review);
            String doc = gson.toJson(anreview);
            RedisClient.addDocument(Const.REDIS_KEY, doc);
        }
    }
```

附录4:集群配置

节点IP	操作系统	CPU	内存
192.168.56.3 (master)	Ubuntu 14.1	4 cores	8GB
192.168.56.4 (slave1)	Mac OS X 10.11	8 cores	16GB
192.168.56.5 (slave2)	Ubuntu 14.1	4 cores	8GB
192.168.56.6 (slave3)	Mac OS X 10.11	8 cores	16GB
192.168.56.7 (slave4)	Mac OS X 10.11	8 cores	16GB
192.168.56.8 (slave5)	Ubuntu 14.1	4 cores	8GB

附录5:开源工具包

开源项目	开发机构	维护组织	版本
Hadoop	Google Inc.	Apache基金会	2.6.2
Spark	University of California, Bekeley	Apache基金会	1.5.2
Stanford CoreNLP	Stanford University, NLP Group	Stanford University	3.5.2
ElasticSearch	Github opensource project	Apache Lucene™	2.1.1
Redis	Redis Labs	Redis Labs	3.2
MongoDB	10GEN	MongoDB Community	3.2