计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20160197
53(8):1819-1828,2016

SparkCRF:一种基于 Spark 的并行 CRFs 算法实现

朱继召[1,2] 贾岩涛[2] 徐 君[2] 乔建忠[1] 王元卓[2] 程学旗[2]
[1](东北大学计算机科学与工程学院 沈阳 110819)
[2](中国科学院计算技术研究所网络数据科学与技术重点实验室 北京 100190)
(zhujzh.paper@gmail.com)

SparkCRF: A Parallel Implementation of CRFs Algorithm with Spark

Zhu Jizhao[1,2], Jia Yantao[2], Xu Jun[2], Qiao Jianzhong[1], Wang Yuanzhuo[2], and Cheng Xueqi[2]
[1](*College of Computer Science and Engineering, Northeastern University, Shenyang* 110819)
[2](*Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

Abstract Condition random fields has been successfully applied to various applications in text analysis, such as sequence labeling, Chinese words segmentation, named entity recognition, and relation extraction in nature language processing. The traditional CRFs tools in single-node computer meet many challenges when dealing with large-scale texts. For one thing, the personal computer experiences the performance bottleneck; For another, the server fails to tackle the analysis efficiently. And upgrading hardware of the server to promote the capability of computing is not always feasible due to the cost constrains. To tackle these problems, in light of the idea of "divide and conquer", we design and implement SparkCRF, which is a kind of distributed CRFs running on cluster environment based on Apache Spark. We perform three experiments using NLPCC2015 and the 2nd International Chinese Word Segmentation Bakeoff datasets, to evaluate SparkCRF from the aspects of performance, scalability and accuracy. Results show that: 1) compared with CRF++, SparkCRF runs almost 4 times faster on our cluster in sequence labeling task; 2) it has good scalability by adjusting the number of working cores; 3) furthermore, SparkCRF has comparable accuracy to the state-of-the-art CRF tools, such as CRF++ in the task of text analysis.

Key words big data; machine learning; distributed computing; Spark; condition random fields (CRFs)

摘 要 条件随机场(condition random fields, CRFs)可用于解决各种文本分析问题,如自然语言处理(natural language processing, NLP)中的序列标记、中文分词、命名实体识别、实体间关系抽取等.传统的运行在单节点上的条件随机场在处理大规模文本时,面临一系列挑战.一方面,个人计算机遇到处理的瓶颈从而难以胜任;另一方面,服务器执行效率较低.而通过升级服务器的硬件配置来提高其计算能力的方法,在处理大规模的文本分析任务时,终究不能从根本上解决问题.为此,采用"分而治之"的

收稿日期:2016-03-21;**修回日期**:2016-06-08
基金项目:国家"九七三"重点基础研究发展计划基金项目(2014CB340405,2013CB329602);国家重点研发计划基金项目(2016YFB1000902);国家自然科学基金项目(61173008,61232010,61272177,61303244,61402442);北京市自然科学基金项目(4154086)
This work was supported by the National Basic Research Program of China (973 Program) (2014CB340405, 2013CB329602), Nationlal Key Research and Development Program of China (2016YFB1000902), the National Natural Science Foundation of China (61173008, 61232010, 61272177, 61303244, 61402442), and the Natural Science Foundation of Beijing (4154086).
通信作者:乔建忠(qiaojianzhong@mail.neu.edu.cn)

思想,基于 Apache Spark 的大数据处理框架设计并实现了运行在集群环境下的分布式 CRFs——SparkCRF. 实验表明,SparkCRF 在文本分析任务中,具有高效的计算能力和较好的扩展性,并且具有与传统的单节点 CRF++相同水平的准确率.

关键词 大数据;机器学习;分布式计算;Spark;条件随机场

中图法分类号 TP181

条件随机场(condition random fields, CRFs)[1]是在已知输入随机变量条件下,输出随机变量的条件概率分布模型. 可以用于文本分析、计算机视觉、生物信息学等众多领域中的预测问题,如自然语言处理中的序列标记、中文分词、命名实体识别,实体间关系抽取等. 随着互联网、物联网、云计算等技术的迅猛发展,网络空间中各类应用层出不穷,引发了数据规模的爆炸式增长[2],预计到 2020 年,全球的数据总量将达到 35 ZB. 传统的运行在单节点上的 CRFs 工具在处理大规模计算任务时(任务的计算规模与训练集大小、模板和输出标记数量有关),面临一系列挑战. 一方面,个人计算机遇到处理的瓶颈从而难以胜任;另一方面,服务器执行效率较低. 在面对大数据[3-4]时代下的大规模计算任务时,通过升级服务器的硬件配置来提高其计算能力的方法,终究不能从根本上解决问题. 因而,将大规模计算任务转移到分布式集群平台上处理,成为了解决以上问题的有效途径. 代表性的批处理系统有 MapReduce[5], Storm[6], Spark[7], Scribe, Samza, Flume, Nutch, Dremel[8], Pregel[9] 和 Trinity[10] 等,其中 Spark 近年来受到了更多用户的青睐并成为了最活跃、最高效的大数据通用计算平台. 这些优秀分布式处理框架,使得开发分布式应用、充分利用集群中的资源变得十分容易,从而大大减小了大规模数据分析任务的难度.

为了解决 CRFs 在处理较大数据集时的低效问题,文献[11]基于多核实现了 CRFs 训练过程的并行化;文献[12]通过将学习过程分解为若干更小、更简单子问题的方式来处理复杂的计算任务;文献[13]基于 MapReduce 计算框架实现了 CRFs 的训练过程并行化,在保证训练结果正确性的同时,大大减少了训练时间,性能上也得到了一定的提升. 然而,由于 MapReduce 自身的实现机制,每次 Mapper 需要将计算结果写入磁盘. 对于训练过程需要大量迭代操作的机器学习算法,这在效率上存在瓶颈,尤其针对大规模训练数据集的计算任务.

针对单机 CRFs 在处理大规模训练数据集时存在的问题. 我们采用“分而治之”的思想,基于 Apache Spark 大数据处理框架设计并实现了运行在集群环境下的分布式 CRFs——SparkCRF. SparkCRF 充分利用了 Spark 提供的弹性分布式数据集(resident distributed datasets, RDD),将所有的数据转化成 RDD 存储在集群节点的内存中,实现了基于内存的分布式计算方式. 最后,通过设计的 3 组实验,分别从执行效率、可扩展性和正确性方面对 SparkCRF 进行评估. 实验表明,SparkCRF 具有高效执行计算任务的能力和较好的可扩展性,并且具有与 CRF++相同水平的预测准确率.

1 背景知识

1.1 条件随机场

CRFs 作为一种统计序列模型首先于 2001 年被 Lafferty 提出[1],至今,已被广泛应用到自然语言处理、计算机视觉、生物信息学等众多领域中. 模型思想的主要来源是最大熵模型,是在给定待标记的观察序列条件下的整个标记序列的联合概率分布,模型中的概率计算、学习和预测 3 个问题的解决和隐马尔可夫模型(hidden Markov model, HMMs)模型类似,使用到的算法如前向-后向和维特比.

CRFs 是一种用来标记和切分序列化数据的概率统计模型. 通常被用于在给定需要标记的观察序列的条件下,计算整个标记序列的联合概率. 标记序列的分布条件属性,可以让 CRFs 很好地拟和现实数据,而在这些数据中,标记序列的条件概率信赖于观察序列中非独立的、相互作用的特征,并通过赋予特征以不同权值来表示特征的重要程度.

条件随机场的定义:设 X 与 Y 是随机变量,$P(Y|X)$是在给定 X 的条件下 Y 的条件概率分布. 若随机变量 Y 构成一个由无向图 $G=(V,E)$表示的马尔可夫随机场(Markov random field, MRF),即 $P(Y_v|X,Y_w,w\neq v)=P(Y_v|X,Y_w,w\sim v)$对任意节点 v 成立,则称条件概率分布 $P(Y|X)$为条件随机场,其中 $w\sim v$ 表示在同 $G=(V,E)$中与节点 v 有边链接的所有节点 w,$w\neq v$ 表示节点 v 以外的所有节点,Y_v,Y_u,Y_w 为节点 v,u,w 对应的随机变量.

条件随机场能够充分地利用上下文信息从而达到良好的标注效果,目前实际应用中最常用的是一阶链式结构的 CRF 模型,即线性链结构(linear-chain CRFs),其结构如图 1 所示:

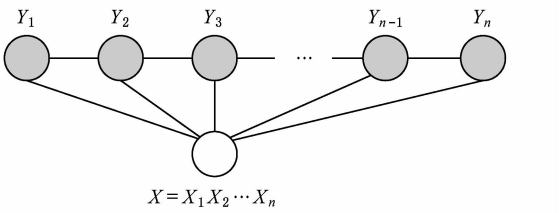


Fig. 1 The structure of linear Chain CRFs.

图 1 线性链条件随机场

条件随机场的参数化形式定义:设 $P(Y|X)$ 为线性链条件随机场,则在随机变量 X 取值为 x 的条件下,随机变量 Y 取值为 y 的条件概率

$$P(y|x)=\frac{1}{Z(X)}\exp\Big(\sum_{i,k}\lambda_k t_k(y_{i-1},y_i,x,i)+\sum_{i,l}\mu_l s_l(y_i,x,i)\Big),$$

其中,

$$Z(X)=\sum_{y}\exp\Big(\sum_{i,k}\lambda_k t_k(y_{i-1},y_i,x,i)+\sum_{i,l}\mu_l s_l(y_i,x,i)\Big),$$

式子中,t_k 和 s_l 是特征函数,λ_k 和 λ_l 是对应的权值,$Z(x)$是规范化因子,求和是在所有可能的输出序列上进行的.t_k 是定义在边上的特征函数,称为转移特征,依赖于当前和前一个位置,s_l 是定义在节点上的特征函数,称为状态特征,依赖于当前位置.t_k 和 s_l 都依赖于位置,是局部特征函数.通常,特征函数 t_k 和 s_l 取值为 1 或 0;当满足特征条件时取值为 1,否则为 0.条件随机场完全由特征函数 t_k,s_l 和对应的权值 λ_k、λ_l 确定.

1.2 Spark 计算框架

Spark[7]是一种基于内存计算的分布式集群计算框架.与 MapReduce 相比具有多方面的改进,即较低的网络传输和磁盘 I/O 使得效率得到提高,Spark 使用内存进行数据计算以便快速处理查询、实时返回分析结果,Spark 提供比 Hadoop 更高层的 API,同样的算法在 Spark 中的运行速度比 Hadoop 快 10~100 倍[7].

Spark 是为集群计算中的特定类型的工作负载而设计,即在并行操作之间重用工作数据集(比如机器学习任务)的工作负载.Spark 的计算架构具有 3 个特点:

1) Spark 拥有轻量级的集群计算框架.它将 Scala 应用于其程序架构,而 Scala 这种多范式的编程语言具有并发性、可扩展性以及支持编程范式的特征,与 Spark 紧密结合,能够轻松地操作分布式数据集,并且可以轻易地添加新的语言结构.

2) Spark 包含了大数据领域的数据流计算和交互式计算.Spark 可以与 HDFS[14]交互取得里面的数据文件,同时 Spark 的迭代、内存计算以及交互式计算为数据挖掘和机器学习提供了很好的框架.

3) Spark 有很好的容错机制.Spark 使用了 RDD,RDD 被表示为 Scala 对象分布在一组节点中的被序列化的、只读的、具有容错机制的并且可被并行执行的对象集合.Spark 高效处理分布数据集的特征使其成为了最活跃、最高效的大数据通用计算平台.图 2 描述了 Spark 的运行模式.

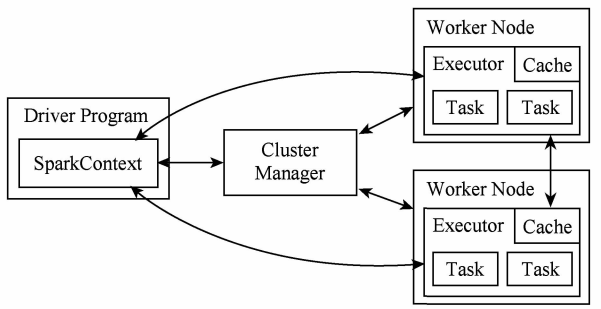


Fig. 2 Operation schema of Spark.

图 2 Spark 运行模式

2 问题提出

在众多的 CRFs 算法工具中,最具代表性的是 CRF++①.CRF++提供了 Linux 和 Windows 2 个版本,是一款开源的用于分词、序列标记等任务的工具.它的设计和实现基于通用性的目的,并已被应用于多种自然语言处理任务.CRF++是单机程序,它能够处理的计算规模与机器的硬件配置关系密切,其中计算规模与训练集大小、模板和输出标记数量有关.这些单机实现工具在处理计算规模较大的任务时,面临一系列挑战.1)由于个人计算机的处理能力有限,在面对较大规模的计算任务时往往难以胜任;2)服务器处理能力较为强大,但在处理大规模计算任务时执行效率却不够高.单纯的通过升级服务器的硬件配置来获得更强的处理能力的方法,面对大数据时代下更大规模的处理任务,其终究不能从根本上解决问题.

① https://sourceforge.net/projects/crfpp/

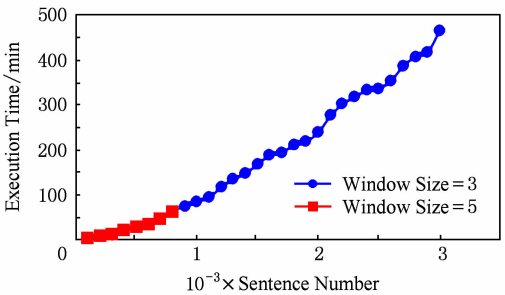
我们以 CRF++ 为例,通过实验来说明以上问题.实验设计如下:1)实验环境:运行环境分别为个人计算机和服务器,配置信息见表 1;2)实验数据:使用 NLPCC2015[①] 公开数据集,选用中文分词数据集 train-POS-EN.txt,该数据集大小共 3 MB,包含 10 000 个句子,输出标记共 122 个;3)模板配置:分别使用窗口大小为 3 和 5 的配置模板用于训练过程中特征抽取.

Table 1 The Configuration of Experiment Platforms

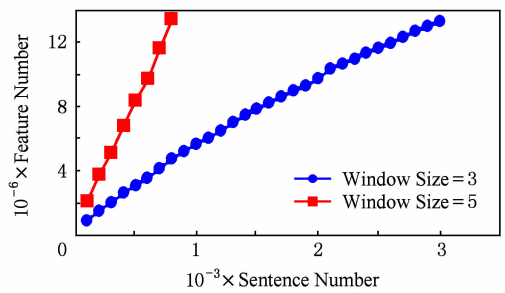
表 1 实验平台配置信息

Item	Personal Computer	Server
Memory/GB	4	128
Hard Disk/GB	500	1×10^3
Processor	Intel® Core™2 2.80 GHz	Intel® Xeon® 2.0 GHz
Core no.	4	24

对于个人计算机,我们对训练集以句子为单位进行切分,切分后的训练集包含的句子总数分别为:100,200,300,400,500,….在每个训练集上按照实验的设计对每个训练过程分别重复进行 5 次,取 5 次的平均执行时间作为最终结果,并统计对应的特征数,实验结果如图 3 所示:



(a) Relationship between execution time and sentence number



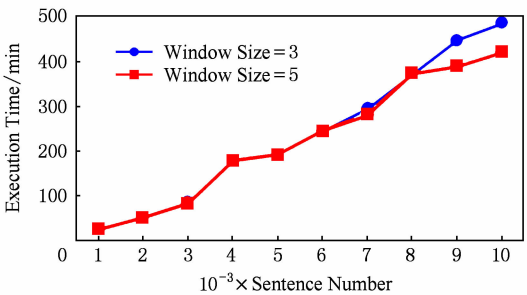
(b) Relationship between feature number and sentence number

Fig. 3 Experiment results on personal computer.

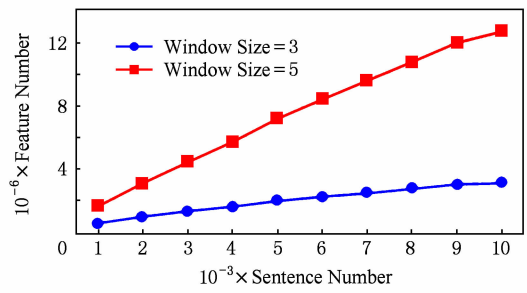
图 3 个人计算机上实验结果

从图 3(a)可以看出:窗口大小为 3 时,能够处理的训练集最多为 3 000 条句子;窗口为 5 时,能够处理的训练集最多为 800 条句子.当训练集大小达到上限后,若继续增加训练句子数量,将会导致系统异常而终止程序运行.出现这种现象的原因从图 3(b)中可知:随着训练集句子规模的增大,对应特征数量也在增长.但对于相同规模的训练集,特征数量却不同,这与所使用模板窗口大小有关.对于不同的窗口设置,当到达机器所能处理的上限时,所对应训练集的特征数基本相同(约 1 400 万).

对于服务器,我们同样按句子对训练集进行切分,切分后的训练集句子总数分别为:1 000,2 000,3 000,4 000,5 000,…,10 000.在每个训练集上按照实验的设计对每个训练过程分别重复进行 5 次,取 5 次的平均执行时间作为最终结果,并统计对应的特征数,实验结果如图 4 所示:



(a) Relationship between execution time and sentence number



(b) Relationship between feature number and sentence number

Fig. 4 Experiment results on server.

图 4 服务器上实验结果

从图 4(a)可知,训练时间随训练集的增大不断增长;当训练集大小从 1 000 增加到 8 000 时,2 种窗口情况下的耗时基本相同.从 4(b)可以看出,随着训练集的增加,特征个数成线性增长;窗口为 5 时的增长速度远快于窗口为 3 时.当训练集增加到 10 000 个句子时,整个训练过程需要 7 h 以上,而此时的训练集却仅仅 3 MB.

实验告诉我们:对于个人计算机,其能够处理的计算规模十分有限;而服务器执行效率较低.通过

① http://nlp.fudan.edu.cn/nlpcc2015/

升级服务器的硬件配置来提高其计算能力的方法,在处理大规模的文本分析任务时,终究不能从根本上解决问题.

根据以上实验分析设想:如果将大的训练集合分割成若干个小的训练集,在多台机器间分布式并行执行,即采用"分而治之"的思想协同工作,那么可处理的训练集大小将会成线性增长.本文以此为动机,将传统的 CRFs 单机运行模式实现为分布式集群运行模式,采用目前分布式计算框架 Spark,以 HDFS[14] 作为数据存储平台.

3 相关工作

从 2001 年 CRFs 被 Lafferty 提出,已被广泛应用到自然语言处理、计算机视觉、生物信息学等众多领域中.到目前为止,很多工作致力于 CRFs 算法工具的实现,最具代表性的是 CRF++.此外,还有 CRF-ADF,GRMM,DGM,CRFall,CRFSuite 等.以上实现存在单机运行模式的共同特点,因此,使用这些工具处理大规模计算任务时必然会面临一系列挑战:1)个人计算机遇到处理的瓶颈从而难以胜任;2)服务器执行效率较低.

文献[11]为了解决 CRFs 的训练过程在规模较大数据集耗时、低效问题,基于多核实现了 CRFs 训练过程的并行化.使得 CRFs 能够处理的训练集规模可达到百上千条序列和百万级别特征;文献[12]通过将学习过程分解为若干更小、更简单子问题的方式,来处理复杂的计算任务;文献[13]通过在 CRFs 模型中引入惩罚项控制非零系数的个数,同时又可避免能够导致大维度参数设置的数字问题.在忽略执行时间因素下,实现了 CRF 用于处理几百个输出标志和多达几十亿特征的训练任务;文献[15]对 CRF 采用平均场近似推理及高斯对势函数,提出一种新的神经网络 CRF-RNN,用作 CNN 的一部分来得到一种具有 CNN 和 CRF 的属性的深度网络应用于图像识别.

文献[16]基于 MapReduce 计算框架实现了 CRFs 的训练过程并行化.该方法确保训练结果正确性的同时,大大减少了训练时间,性能也得到了一定的提升.由于 MapReduce 自身的机制,每次 Map 完成需要将操作结果存入磁盘,这显然对于训练过程需要大量迭代操作的机器学习类算法在实现效率上存在瓶颈.因此,这种基于 MapReduce 的实现虽然能够处理较大规模的训练集和特征集,但运行效率并不高.

文献[17]基于 Spark 提出一种并行蚁群优化算法,相比于基于 MapReduce 平台下的实现,在速度上提升 10 倍以上;文献[18]基于 Spark 实现了并行化频繁项集挖掘算法,并在多项基准实验上与基于 MapReduce 的实现算法进行对比,实验证明基于 Spark 的频繁项集挖掘算法比基于 MapReduce 的实现在运行速度上平均提升了 18 倍.语义数据在大数据时代下快速增加,传统的单机语义推理实现工具的处理能力面临了严峻的挑战;文献[19]在 Spark 框架上实现了一种高效的大规模 RDFS/OWL 推理工具 Cichlid,并具有良好的可扩展性和容错能力.

4 SparkCRF 的实现

本节描述 SparkCRF 的实现过程,在工作流程上它包含 2 个阶段:训练和预测.SparkCRF 在集群中通过创建 Driver/Executor 进程实现分布式运行,对输入数据格式的要求与 CRF++一致.

4.1 训练阶段

训练阶段包含数据加载、特征抽取和循环迭代调整特征权重的训练过程,最终生成训练模型.

该阶段所用到特征模板、训练集数据预先存储在 HDFS 中,将数据转换成 Spark RDD 形式存储到集群的内存,图 5 描述了通用的加载过程.

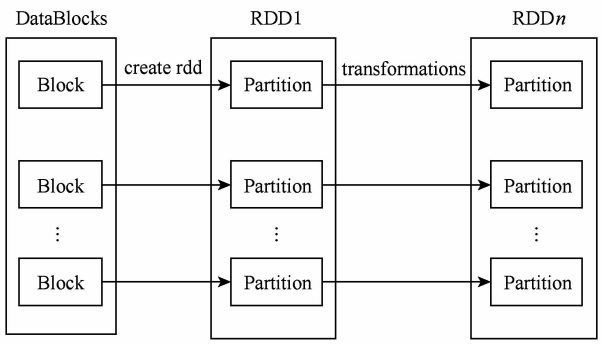


Fig. 5 The general phase of loading data.

图 5 数据加载通用过程

特征模板和训练集数据以 Block 形式存储在 HDFS 系统中,由 SparkContext 对象通过 textFile 方法将数据转化为以 Partition 形式存储的 RDD 并加载到内存中.对于特征模板 RDD 调用 filter 并传入规则,将无效数据过滤,然后分别获取 U 和 B 模板.训练集 RDD 通过一系列转换(transformations)

操作进行处理,将原始数据以句子为单位根据自定义的类型转化为句子封装类,每个 Partition 中包含若干个封装类单元存储在集群节点的内存中.

算法 1. 训练过程伪代码.

输入:训练集 RDD、SparkContext 对象、迭代最大次数和收敛条件;

输出:特征集合、权值向量、模型元数据.

```
Loop:
  Executor:
    对每个包装实例:
      重置图中所有 node 和 edge 对象的 cost 值;
      ForwardBackward();
      Viterbi();
  Driver:
    更新期望;
    调用 L-BFGS 算法模块更新权值向量;
End Loop
```

算法 1 描述了训练过程的总体流程.训练集 RDD,SparkContext 对象、训练过迭代最大次数及收敛值作为过程的输入;这里以 NLP 中序列标记为例进行描述,训练过程开始先将特征模板 RDD 广播到各个工作节点.在各工作节点上,对训练集以句子为单位进行特征抽取并转化成一个句子包装类别实例,这样训练集 RDD 被转化为新的 RDD 存储在内存中.新的 RDD 中的每个实例对象包含有 $m \times n$ 大小的图,其中 m 和 n 分布代表句子中含有的字个数和可能的输出标记个数.Driver 进程根据抽取的特征集大小初始化对应的权重数组.训练迭代过程开始,每个句子包装对象中:1)重置图中节点对象和边对象的 *cost* 值;2)调用 ForwardBackward 函数(见算法 2)计算图中每个节点的 α 和 β 值.前向算法是给定到特定观测及该观测之前的所有连续观测序列,计算该观测下特定状态出现的概率,后向算法是给定到特定观测及该观测之后的连续观测序列,计算该观测下特定状态出现的概率.3)计算每个特征的期望改变量,以〈*index*,*incremental_value*〉对的形式临时存储下来,待解码过程完返回给 Driver 进程;4)解码过程调用函数 *Viterbi*(见算法 3)实现.Driver 进程首先更新期望数组的值,然后使用 L-BFGS 算法对权值数组进行更新.每次迭代过程结束,判断是否满足终止训练的条件.训练过程结束后,输出特征的权值向量、特征集合和模型的元数据信息.

算法 2. ForwardBackward 实现过程伪代码.

```
For num_r=1 to 句子长度 total_length do
  For num_c=1 to 输出标记数 tags_number do
    计算图中第 num_r 行,第 num_c 列节点的前向概率 α;
    计算图中第 total_length-num_r 行,第 num_c 列节点的后向概率 β;
    计算当前节点所以输出边 edge 的 cost 值;
  End For
  计算规范化因子 Z;
End For
```

算法 3. 训练 Viterbi 实现过程伪代码.

```
For num_w=1 to 句子长度 senten_length do
  For num_t=1 to 输出标记数 tags_number do
    初始化变量 best_cost, best_node;
    获取第 num_w-th 行、第 num_t-th 列的节点,用 node_w_t 表示;
    遍历节点 node_w_t 的所有输出边 edge;
    对节点的所有输入边:
      计算边的总代价 cost,当前节点 node_w_t 的 cost 及前向连接节点中最小的 cost 的节点;
      IF cost>best_cost
        更新 best_cost;
        更新 best_node;
      END IF
    更新节点 node_w_t 的前驱指针 prev;
    更新节点 node_w_t 的 best_cost;
  End For
End For
Return 最优的路径,即输出标记序列.
```

每轮迭代过程,Driver 进程与 Executor 进程通信 2 次:1)广播特征权值向量;2)接受各个 Executor 返回的期望增量集合和局部的梯度改变量.数据的传递和接受基于 Spark 内在操作机制完成.

4.2 预测阶段

Driver 进程首先从 HDFS 上读取训练好的模型数据和测试集并转化为 RDD 存储在内存中,将模型 RDD 广播到各个工作节点的 Executor 进程.Executor 进程执行特征抽取操作,重置图中节点对象和边对象的 *cost* 值,通过 ForwardBackward 算法完成路径代价的计算,最后使用维特比算法解码找出最优的输出标记序列.

算法 4. 预测过程伪代码.

```
Driver:
    从HDFS 加载训练模型和测试集,转化为
      RDD;
    将模型广播到工作节点;
Executor:
    对测试集 RDD 进行 map 操作;
    对每个包装对象:
      生成特征;
      重置图中所有 node 和 edge 对象的 cost 值;
      ForwardBackward();
      Viterbi();
输出预测序列.
```

5 实验结果与分析

实验在我们的集群服务器上进行,共有 8 台配置相同的服务器节点组成(配置信息见表 1),其中主节点 1 台、从节点 7 台. 集群服务器操作系统为 CentOS 6.6,安装有 JDK 1.8.0,SparkCRF 的设计和实现基于 Spark-1.5.2. 其中,HDFS 与 Spark 集群安装在同一集群服务器. 实验使用到的数据均已存储到 HDFS 中,并且数据格式与 CRF++的输入数据要求一致.

我们设计了 3 组实验,分别从执行效率、可扩展性和正确性方面对 SparkCRF 性能进行评估. 由于预测阶段是一个使用训练过程生成的模型对测试数据进行解码的过程,各个工作节点在获得 Driver 进程广播的数据后,整个过程均在各工作节点的 Executor 进程中执行,Executor 间不存在数据的传输. 因此,预测阶段执行时间与参与工作的节点个数成正线性关系. 因此,高效性和可扩展性的验证,我们只对训练阶段进行.

5.1 高效性

为了验证 SparkCRF 比单机 CRF 算法工具更加高效,我们继续选用 NLPCC2015 公开数据集中的 train-POS-EN.txt 数据,将其切分为句子条数为 1 000,2 000,3 000,…,10 000 的 10 份数据分别用于训练,单机程序继续使用 CRF++. 考虑到单机下的训练过程十分耗时,我们复用第 2 节的实验数据. 我们将训练数据集分别在集群环境下使用 SparkCRF 运行,实验在 Spark 集群可获取的最大服务器资源下进行. 当模板窗口大小为 3 时,实验结果如图 6(a)所示,窗口大小为 5 时,结果如图 6(b)所示.

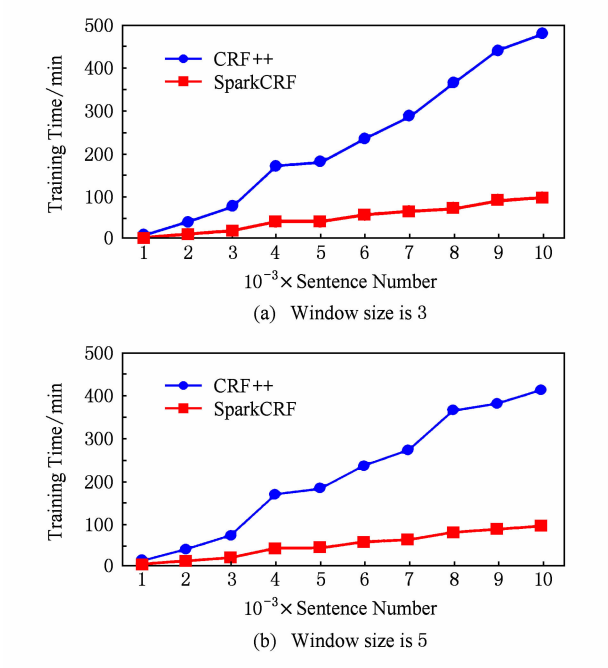


Fig. 6 CRF++ vs SparkCRF.

图 6 CRF++与 SparkCRF 对比结果

从图 6 描述的实验结果可知:1)CRF++和 SparkCRF 的执行时间均随着训练集的增大而增加,并呈现正线性关系;2)在相同的训练集上,SparkCRF 消耗时间大约为 CRF++耗时的 $\frac{1}{4}$. 因为 SparkCRF 运行过程中,集群环境下提供了更多的计算资源,而 CRF++仅使用一台服务器资源. 然而,集群环境下 Executor 进程与 Driver 进程每轮迭代结束存在数据传输以及部分计算任务需要在 Driver 独立完成. 因此,在我们的集群环境下,SparkCRF 的执行时间与 CRF++相比并没能减少到所增加计算资源的倍数.

5.2 可扩展性

我们的实验在第 2 届国际中文分词公开数据集[①]上对 SparkCRF 进行可扩展性测试. 公开的数据集包含 4 套用于训练和测试的数据,它们分别由微软亚洲研究院(Microsoft Research Asia, MSR)、香港城市大学(City University of Hongkong, CityU)、台湾中央研究院(Academia Sinica, AS)和北京大学(Peking University, PKU)提供. 根据我们的了解,这份数据集是目前可免费获取到的最大的中文分词数据集,由于其在数据格式与 SparkCRF 对输入

① http://www.sighan.org/bakeoff2005/

数据的格式要求不一致,因此我们开发一个数据格式转换程序对其进行预处理.训练数据的详细统计信息见表2所示,其中WT代表Word Types,SN代表Sentences Number,PFS代表Processed File Size.

实验在仅处理器核数为可变因素、其他所有因素均不变的条件下进行.我们将SparkCRF的工作核数分别设定为50,70,90,110,130和150来验证其可扩展性能.对每个训练数据,分别在指定的每种处理器核数下,对同一实验重复进行5次,记录下运行时间,最后对这5次结果取平均值作为执行时间.

实验结果如图7所示.

Table 2 Statistical Information of the Second International Chinese Word Segmentation Bakeoff

表2 第2届国际中文分词数据集统计信息

Datasets	#WT	#Word	#SN	#Char	#PFS/MB
MSR	8.8×10^4	2.4×10^6	8.7×10^4	4.1×10^6	23.2
CityU	6.9×10^4	1.5×10^6	5.3×10^4	2.4×10^6	13.6
AS	14.1×10^4	5.4×10^6	70.9×10^4	8.4×10^6	48.5
PKU	5.5×10^4	1.1×10^6	1.9×10^4	4.7×10^6	10.4

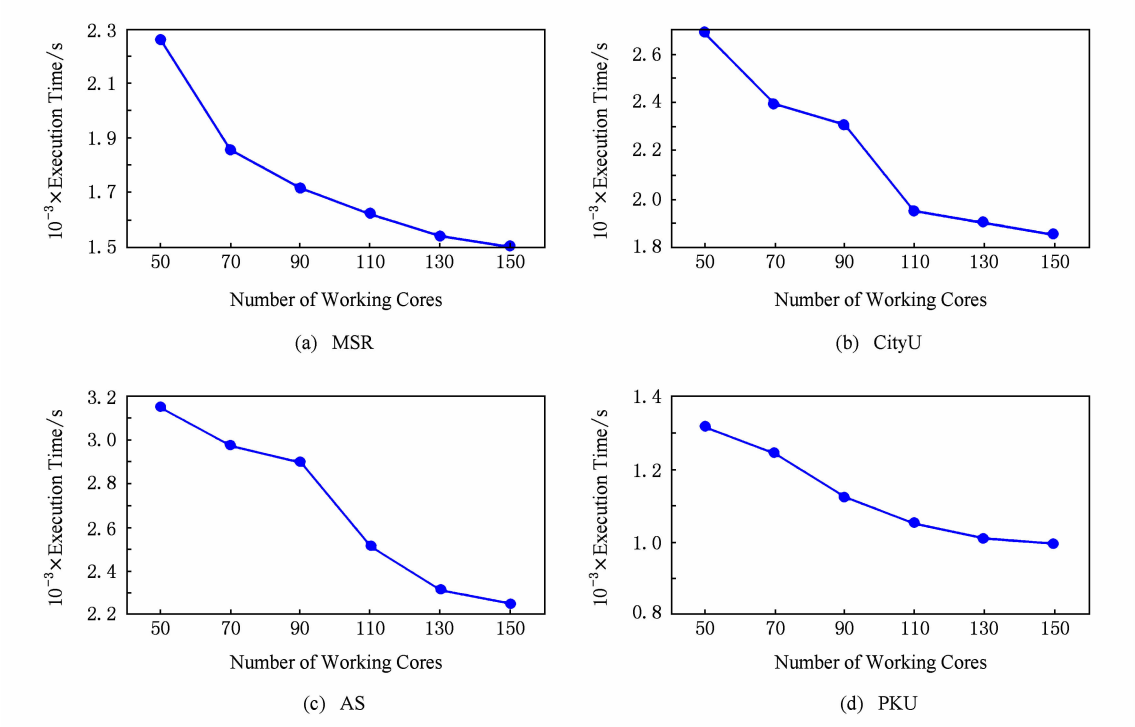


Fig. 7 Relationship between core number and execution time on different datasets.

图7 不同数据集上核数与执行时间的关系

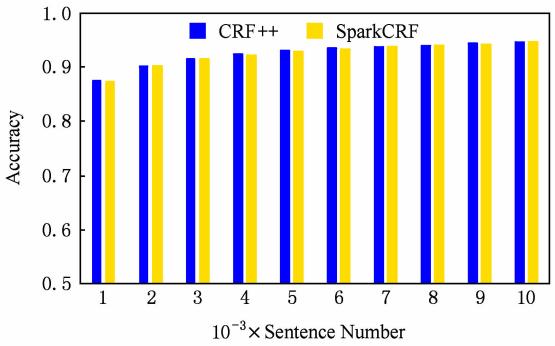
图7(a)(b)(c)(d)分别描述了在训练数据集MSR,CityU,AS和PKU上的实验结果.我们从图7中发现:1)随着工作核数的增加,不同数据集上的训练时间均在减少;2)不同数据集间的表现虽有一定的差别,但总体上工作核数与执行时间均呈现出了线性的关系.

5.3 正确性

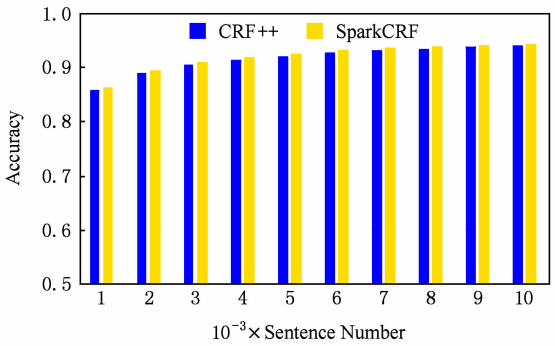
为了检验SparkCRF的正确性,我们选择与CRF++进行实验对比.实验训练数据集使用NLPCC2015公开数据集中的train-SEG.txt,预测数据集为对应的test-Gold-SEG.txt.通过2组实验进行验证,特征模板窗口大小分别设置为3和5,训练集包含的句子数分别为1 000,2 000,3 000,…,10 000.

按照实验的设计,1)分别使用CRF++和SparkCRF在训练集集合上进行训练,分别生成的模型;2)使用生成的模型分别用于对标准测试集进行预测;3)统计预测的准确率.图8(a)展示了窗口大小为3时的结果,图8(b)为窗口大小为5时的结果.

实验结果表明:1)2种窗口大小下,CRF++和SparkCRF预测的准确率均随着训练规模的增大而提高;2)不同窗口下,二者在相同训练集上的准确率稍有差别.窗口为3时,SparkCRF略低于CRF++的准确率,窗口为5时,SparkCRF预测的准确率稍高于CRF++,但总体上二者的预测准确率基本相同;3)窗口为3时,当训练集句子增加到8 000后,此时CRF++和SparkCRF的准确率分别为94.12%



(a) Window size is 3



(b) Window size is 5

Fig. 8 Relationship between sentence number of corpus and accuracy.

图 8 训练集句子个数与预测正确率的关系

和 94.14%,增加训练集包含的句子数对该测试集预测准确率的提高不明显;窗口为 5 时,当训练集增加到 9 000 条句子时,CRF++/SparkCRF 对应的准确率为 93.73%/94.04%,10 000 条句子时对应的准确率为 94.07%/94.33%.通过分析可知,模型中包含的该测试集所需特征的比重是影响预测准确率的根本因素.

6 总 结

针对传统的运行在单节点上的条件随机场在处理大规模文本时,面临一系列挑战.本文采用"分而治之"的思想,基于 Apache Spark 的大数据处理框架设计并实现了运行在集群环境下的分布式 CRFs.通过对文本分析任务进行的实验,证明了 SparkCRF 具有高效的计算能力和较好的扩展性,并且具有与传统的单节点 CRF++相同水平的准确率.

在理论上,SparkCRF 能够处理大规模的数据集.由于可获得标准训练集有限,我们仅对几十兆级大小的数据集进行了测试.因此,还有待进一步在大规模数据集上对 SparkCRF 验证;在应用方面,利用 SparkCRF 工具提升文本处理的效果也是一个有待尝试的工作.例如,与知识表示[20]工作相结合,提升领域知识的抽取效果.以上两个方面也是我们未来的工作重点.

参 考 文 献

[1] Lafferty J, McCallum A, Pereira F C N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data [C] //Proc of the Int Conf on Machine Learning. Francisco, CA: Morgan Kaufmann, 2001: 282-289

[2] Wang Yuanzhuo, Jia Yantao, Liu Dawei, et al. Open Web knowledge aided information search and data mining [J]. Journal of Computer Research and Development, 2015, 52(2): 456-474 (in Chinese)
(王元卓, 贾岩涛, 刘大伟, 等. 基于开放网络知识的信息检索与数据挖掘[J]. 计算机研究与发展, 2015, 52(2): 456-474

[3] Wang Yuanzhuo, Jin Xiaolong, Cheng Xueqi. Network big data: Present and future [J]. Chinese Journal of Computers, 2013, 36(6): 1125-1138 (in Chinese)
(王元卓, 靳小龙, 程学旗. 网络大数据:现状与挑战[J]. 计算机学报, 2013, 36(6): 1125-1138)

[4] Cheng Xueqi, Jin Xiaolong, Wang Yuanzhuo, et al. Survey on big data system and analytic technology [J]. Journal of Software, 2014, 25 (9): 1240-1252 (in Chinese)
(程学旗, 靳小龙, 王元卓, 等. 大数据系统和分析技术综述[J]. 软件学报, 2014, 25(9): 1889-1908.)

[5] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113

[6] Toshniwal A, Taneja S, Shukla A, et al. Storm@Twitter [C] //Proc of the 2014 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2014: 147-156

[7] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets [C] //Proc of the 2nd USENIX Workshop on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2010: 10-10

[8] Melnik S, Gubarev A, Long J J, et al. Dremel: Interactive analysis of web-scale datasets [J]. Proceedings of the VLDB Endowment, 2010, 3(1/2): 330-339

[9] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing [C] //Proc of the 2010 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 135-146

[10] Shao Bin, Wang Haixun, Li Yatao. Trinity: A distributed graph engine on a memory cloud [C] //Proc of the 2013 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2013: 505-516

[11] Phan H X, Nguyen M L, Horiguchi S, et al. Parallel training of CRFs: A practical approach to build large-scale prediction models for sequence data [C] //Proc of Int Workshop on Parallel Data Mining. Berlin: Springer, 51-63

[12] Bradley J K. Learning large-scale conditional random fields [D]. Pittsburgh: Carnegie Mellon University, 2013

[13] Lavergne T, Cappé O, Yvon F. Practical very large scale CRFs [C] //Proc of the 48th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2010: 504-513

[14] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system [C] //Proc of the 2010 IEEE 26th Symp on Mass Storage Systems and Technologies. Piscataway, NJ: IEEE, 2010: 1-10

[15] Zheng Shuai, Jayasumana S, Romera-Paredes B, et al. Conditional random fields as recurrent neural networks [C] // Proc of the IEEE Int Conf on Computer Vision. Piscataway, NJ: IEEE, 2015: 1529-1537

[16] Liu Tao, Lei Lin, Chen Luo, et al. A parallel training research of Chinese words part-of-speech tagging CRF model based on MapReduce [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2013 (1): 147-152 (in Chinese)
(刘滔, 雷霖, 陈荦, 等. 基于 MapReduce 的中文词性标注 CRF 模型并行化训练研究[J]. 北京大学学报: 自然科学版, 2013 (1): 147-152)

[17] Wang Zhaoyuan, Wang Hongjie, Xing Huanlai, et al. Ant colony optimization algorithm based on Spark [J]. Journal of Computer Application, 2015, 35(10): 2777-2780, 2797
(王诏远, 王宏杰, 邢焕来, 等. 基于 Spark 的蚁群优化算法 [J]. 计算机应用, 2015, 35(10): 2777-2780, 2797)

[18] Qiu Hongjian, Gu Rong, Yuan Chunfeng, et al. YAFIM: A parallel frequent itemset mining algorithm with Spark [C] // Proc of the 2014 IEEE Int Conf on Parallel & Distributed Processing Symp Workshops. Piscataway, NJ: IEEE, 2014: 1664-1671

[19] Gu Rong, Wang Shanyang, Wang Fangfang, et al. Cichlid: Efficient large scale RDFS/OWL reasoning with Spark [C] // Proc of the 2015 IEEE Int Conf on Parallel and Distributed Processing Symposium. Piscataway, NJ: IEEE, 2015: 700-709

[20] Jia Yantao, Wang Yuanzhuo, Lin Hailun, et al. Locally adaptive translation for knowledge graph embedding [C] //Proc of the 30th AAAI'6. Palo Alto, CA: AAAI, 2016: 992-998

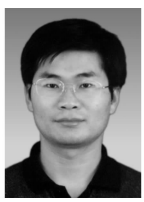
Zhu Jizhao, born in 1986. PhD candidate. His main research interests include open knowledge network, representation learning and parallel computing.

Jia Yantao, born in 1983. PhD, assistant professor. His main research interests include open knowledge network, social computing and combinatorial algorithm.

Xu Jun, born in 1979. Professor, PhD supervisor. His main research interests include information retrieval, machine learning and big data analysis.

Qiao Jianzhong, born in 1964. Professor, PhD supervisor. His main research interests include operation system, parallel and distributed computing.

Wang Yuanzhuo, born in 1978. PhD, associate professor. IEEE member and senior member of China Computer Federation. His main research interests include social computing, open knowledge network, network security analysis, stochastic game model, etc.



Cheng Xueqi, born in 1971. Professor, PhD supervisor. His main research interests include network science, Web search & data mining.