

基于大数据平台的中文文本分析系统研究与实现

摘 要

进入 21 世纪, 互联网技术的发展突飞猛进, 互联网已经与各行各业息息相关, 同时, 连接在互联网上的各种设备在源源不断产生着数据, 造成数据的爆炸式增长, 这其中就包括大量的文本信息。这些文本信息以日志、评论、文章等形式呈现在互联网上, 由于互联网与人们的生活越来越紧密, 网络对社会热点的影响也越来越大, 如何分析网络观点、预测网络情绪并正确引导网络舆情成为当今社会乃至全世界亟需解决的问题, 其中对文本分析技术的研究是解决这一问题的关键点。但是, 现阶段的文本分析方法大都是使用统计语言模型对文本建模, 再结合机器学习算法训练模型, 其效果依赖语料质量, 可复用性差, 而且模型训练需要耗费大量计算时间, 在算法的并行化研究方面缺乏可行的解决方案。为此, 本文以神经网络语言模型为基础, 并结合 Spark 大数据平台, 设计并实现一套中文文本分析处理的综合系统。

本文主要工作包括: (1) 研究基于神经网络语言模型的文本倾向分析算法, 并设计了融合 LDA 模型的 doc2vec 文本特征表示算法。(2) 研究文本分析算法的并行化技术, 利用 Spark 大数据平台设计算法的并行化模型。

(3) 研究中文文本倾向分析流程, 设计并实现基于大数据平台的中文文本分析系统, 包括语料摄取、语料标注、语料存储、模型训练、模型验证等模块。(4) 对原型系统进行验证、测试与分析。

为了验证本课题的可行性, 本文通过对原型系统中设计的融合 LDA 模型的 doc2vec 文本特征表示算法进行准确度测试, 实验结果表明, 经过融合后的文本表示模型, 具有很高的辨识度, 其 ROC 曲线的 AUC 值达到 0.95。同时, 本文对系统涉及的文本分析相关算法进行并行化测试, 测试结果显示, 并行化后的算法可以大大提高系统的效率。

关键词: Spark 文本倾向分析 doc2vec LDA 神经网络 机器学习

RESEARCH AND IMPLEMENTATION OF CHINESE TEXT ANALYSIS SYSTEM BASED ON BIG DATA PLATFORM

ABSTRACT

In the 21st Century, with the rapid development of Internet technology, the Internet has is closely related to all walks of life, forming a pattern of Internet plus. At the same time, a variety of devices connected to the Internet generate data continuously, resulting in a large explosion of data, which laid a foundation for the arrival of the era of big data. This includes a large amount of text information that are presented in the Internet in the form of log, comments, articles and other forms on. Because the Internet and people's lives are getting closer and closer, and the impact of online public opinion on social hot spots is also growing, how to analyze the network view, forecast the network sentiment and correctly guide the network public opinion becomes the problem that needs to be solved urgently in the society and the world. However, the text analysis method at present is to use text modeling of statistical language model, combined with machine learning algorithm to train the model, the effect depends on data quality, and model training needs a lot of computing time, but it is lack of feasible solutions in parallel algorithm research. To this end, this paper based on neural network language model, and combined with Spark big data platform, the design and implementation of a comprehensive system of Chinese text analysis and processing.

The main work of this paper includes: (1) research the text orientation analysis algorithm based on neural network language model, and design the text feature representation algorithm fused doc2vec model with LDA model. (2) research on how to parallel the algorithms which are related with this system, and design the parallel model use the Spark platform. (3) research the general flow of Chinese text orientation analysis system, design and implement the Chinese text analysis system based on big data platform, including the data intake, corpus tagging, corpus storage, model training, model validation etc.(4) verify and test the prototype system, and give the test results.

In order to verify the feasibility of this project, this paper said the algorithm accuracy test of doc2vec text feature fusion LDA model design of the prototype system, the experimental results show that after the fusion of the text representation model, it has a very high degree of recognition, the ROC curve of AUC value reached 0.95. At the same time, this paper makes a parallel test on the text analysis correlation algorithm. The test results show that the parallel algorithm can greatly improve the efficiency of the system.

KEY WORDS: Spark, text orientation analysis, doc2vec, LDA, neural network , machine learning.

目 录

第一章 绪论	1
1.1 研究背景及意义	1
1.2 国内外研究现状	3
1.2.1 文本倾向分析研究综述	3
1.2.2 基于大数据平台的文本分类综述	4
1.3 本文的主要工作	5
1.3.1 研究内容	5
1.3.2 创新之处	5
1.4 本文的组织结构	6
第二章 中文文本分析算法及关键技术	8
2.1 文本分析相关知识点介绍	8
2.1.1 语言模型及文本表示方法	8
2.1.2 文本特征及特征抽取	11
2.2 基于主题模型和文本向量的文本倾向分析算法	13
2.2.1 基于 LDA 的文本特征提取算法	13
2.2.2 基于 word2vec 的文本表示算法	16
2.3 文本分析的关键技术	18
2.3.1 语料采集关键技术	18
2.3.2 文本切分关键技术	19
2.3.3 文本标注关键技术	19
2.4 本章小结	20
第三章 基于大数据平台的文本处理技术	21
3.1 Spark 大数据处理平台	21
3.2 Spark 大数据平台的优势	22
3.3 如何使用 Spark 大数据平台处理文本数据	24
3.4 本章小结	25
第四章 基于大数据平台的文本倾向分析系统的设计	27
4.1 文本倾向分析系统总体框架设计	27
4.1.1 文本预处理模块设计	29
4.1.2 文本存储模块设计	33
4.1.3 文本分析模块设计	35
4.2 基于 doc2vec 和 LDA 的评论文本倾向分析算法设计	35
4.2.1 基于 LDA 的文本主题特征提取	36

4.2.2	结合主题特征的 doc2vec 模型训练.....	37
4.2.3	基于 SGD 的文本倾向分类.....	41
4.3	文本倾向分析算法的并行化实现.....	41
4.3.1	针对文本分词处理的并行化实现.....	41
4.3.2	针对 LDA 算法的并行化实现.....	43
4.3.3	针对 Doc2vec 算法的并行化实现.....	45
4.4	本章小结.....	49
第五章 基于大数据平台的文本倾向分析系统实现及验证测试.....		51
5.1	系统各模块的实现.....	51
5.1.1	文本预处理模块实现.....	51
5.1.2	文本存储模块实现.....	55
5.1.3	文本分析模块实现.....	55
5.2	系统环境搭建与系统部署.....	60
5.3	核心功能模块展示.....	63
5.3.1	爬虫核心模块.....	63
5.3.2	语料生成模块.....	65
5.3.3	文本训练模块.....	66
5.4	系统测试与验证.....	67
5.4.1	准确度测试.....	67
5.4.2	性能测试.....	71
5.5	本章小结.....	73
第六章 结束语.....		75
6.1	论文总结.....	75
6.2	下一步研究工作.....	75
参考文献.....		77

第一章 绪论

1.1 研究背景及意义

进入 21 世纪以来,互联网已经成为人们生活工作中不可缺少的工具,人类进入了信息化时代,在过去的 10 年里计算机为人类提供了巨大的生产力,提高了生产效率,随着计算机科学的发展,尤其是互联网的发展,每天产生的数据量以指数级的速度增长,尤其是近几年社交网络、电子商务平台、自媒体的崛起,使得人们对数据的重视程度越来越高,数据已经成为新型的生产资料被人们发掘、利用,并且在实际的生成环境中带来效益,人类已经不可避免的从信息化时代步入了大数据时代,但是大数据时代的到来既给我们带来机遇也给我们带来了挑战,如何对海量数据进行存储、过滤、处理和分析一直是工业界和学术界有待解决的问题,尤其在大数据中 80%的数据是非结构化的文本数据,其中包括以社交媒体为代表的微博、博客、日志,以虚拟交易平台为代表的商品描述、用户评论,以新闻媒体为代表的时事新闻、讨论等海量文本信息,通过对这些信息的处理和分析可以实现情感分析、舆情分析、商品推荐、金融预测、用户行为描述、广告精准投放等具有实用价值的应用。本文针对互联网上的文本类型数据,设计并实现一种基于大数据平台的中文文本分析系统,旨在为需要进行海量文本分析,尤其是中文文本倾向分析的用户提供一种可行的解决方案。

大数据对于全球还是比较新的课题,各个国家都有自己的大数据研究项目,尤其是以 Google 为代表的新型互联网公司,对大数据技术的探索和发展产生了推动的作用,Google 作为搜索引擎巨头公司,在 2003 年开始陆续发表了三篇有关大数据的论文,分别论述了 GFS [1]、MapReduce[2]、Bigtable[3]三大技术,这三篇论文揭开了大数据技术的神秘面纱,成为工业界对大数据理论的实现标杆。

同时,受到 Google 的启发,Apache 基金会于 2005 年秋天引入了 Hadoop[4]项目,Hadoop 作为开源项目和其稳定的架构设计迅速成为了大数据平台的佼佼者,可以说 Hadoop 已经成为了大数据的代名词,其最大的核心就是实现了一个高容错性的分布式文件系统 HDFS[5]和 MapReduce 计算模型,其中前者为海量数据存储带来解决方案,后者为海量数据计算提供了支持。但是随着 Hadoop 在工业界的广泛使用,其为企业带来便利的同时也暴露出自身的一些局限性,例如抽象层次低、上手难度大;只提供 map 和 reduce 两种操作,表达力不足;复杂的计算需要多个 mapreduce 过程,job 之间的依赖关系由开发者自己管理;时延高,只适合计算密集型任务;对于迭代式数据处理性能差等。

正在大家努力寻求可行的解决方案的时候,Spark 作为新一代的大数据框架正悄悄

地走进人们的视线，Spark 是 UC Berkeley AMP lab 开源的类 Hadoop MapReduce 的通用并行框架，其不但补充了 Hadoop 的不足，而且还集成了很多优秀的组件，例如其增加了数据库和数据框架库、流处理库、机器学习库和图形库。通过引入这些库，用户可以执行更加复杂的任务，并且 spark 将中间结果保存到弹性分布式数据集（Resilient Distributed Datasets RDD）[6]上，该数据集是对分布式内存的抽象映射，这使得 spark 的效率非常高，可以想象 spark 未来发展的前景是非常乐观的。但是，由于 Spark 技术出现的比较早，人们对 spark 的应用仍然处在探索阶段，如何利用 spark 解决现有的大数据问题仍然值得我们去研究和探索。

文本分析相对来说发展的比较成熟，现有的文本分析算法也非常多，效果也经历了工业界的考验，文本分析是自然语言处理发展的必经之路，首先对于自然语言来说，文字是表达语言的重要形式之一，计算机如何正确的理解文本内容，就相当于其可以正确的理解自然语言，在文本分析的初期，计算机理解文本的方式是通过词典映射，间接的“理解”其意思，其实是通过最大字符串匹配获得结果[7][8]，这种方式其实是没有解决计算机理解自然语言的根本问题，随着人们不断的研究，基于概率统计的算法被应用到文本分析领域[9]，并且产生了很好的效果，文本分析进入了新的阶段，近年来机器学习、深度发掘等新的领域发展迅速，人们又找到了文本分析新的突破口，现阶段文本分析的主要思路是结合词典和概率统计算法对语料进行初级处理，之后利用机器学习分类算法对其进行训练，最终得到比较理想的结果[10][11][12][13]。

但是基于机器学习的文本分类算法对训练文本的依赖性很高，而且需要对文本的特征进行提取，不同的特征提取方式会导致结果的精确度不同，常用的特征提取方式是采用 TF-IDF 作为文本的特征值进行训练，效果虽然可以满足要求，但是 TF-IDF 算法没有考虑文本之间上下文语境，准确度仍然有待提高，随着神经网络领域的发展，这时又提出基于神经网络的语言模型[14]，其中郑晓庆博士等人通过利用神经网络模型解决中文分词和词性标注任务[15]，并得到了不错的结果。本文采用了 doc2vec 模型[16]作为文本倾向分析任务的主要模型，并结合 LDA 主题模型[17]对文本特征进行提取，产生了不错的效果。

同时，处理海量的文本信息，需要进行大量的计算任务，如何将这些任务并行化是文本需要解决的另一个问题，通过并行化的方法解决文本处理问题也是非常热的研究方向[18]，其中有学者通过借助 Hadoop 平台处理针对微博文本的情感分类问题[19]，还有通过 Spark 平台解决论文相似性检测的问题[20]，本文通过利用 Spark 大数据平台，对文本进行并行化分析和处理，其中包括对文本的切分，存储，训练和分类等文本分析涉及的过程。

1.2 国内外研究现状

文本分析是自然语言处理过程中最重要的环节，其中文本分析任务最首要的难题是如何表示文本，让计算机更好的理解文本。经过研究者的不断努力，文本表示算法已经有了突飞猛击的发展，从最初的基于词典的表示，到基于上下文的表示，最后到基于词向量的表示，文本表示算法已经满足现阶段的需求。其次需要解决的问题是如何获取文本特征，由于文本表示的特征向量往往具有高维性和稀疏性，这样的表示方法增加了处理文本的难度和成本，如何从高维特征中提取出最具代表性的特征就是文本分析需要解决的第二个难题。最后是对特征表示后的文本进行分类，完成任务需求，分类任务相对于比较容易，因为基于机器学习的分类算法已经非常成熟，但是如何使用特定的分类算法需要根据具体的需求来分析，这样才能达到理想的效果，并且进行算法模型的训练时间往往比较长，如何提高训练效率也是需要考虑的问题。

针对以上提到的三个问题，本文以文本倾向分析为切入点来阐述文本分析领域研究现状。主要通过两个方面进行综述：一、文本倾向分析研究综述，二、基于大数据平台的文本分类。

1.2.1 文本倾向分析研究综述

文本倾向分析是指通过对文本的处理获得该文本的正负情感倾向性，文本倾向分析是文本分析的重要应用领域，通过对文本倾向分析的研究可以应用到整个文本分析过程中涉及的关键技术和重要算法，文本倾向分析概念的形成经历了一个漫长的探索过程，最初是以“意见挖掘(opinion mining)”的概念在 2003 年被 Dave 发表在 www 大会上的论文[21]提出，其认为理想的意见挖掘工具是可以处理一个给定主题的搜索结果集，生成一个产品属性（质量，功能等）的列表，并汇总他们每个人的意见（褒义，中立，贬义）。与“意见挖掘”同时期出现的概念还有“情感分析(sentiment analysis)”，情感分析是由 Das、Chen[22]以及 Tong[23]在 2001 年发表的论文中提出的概念，用来形容在市场情绪分析领域，自动分析预估性文本和预测性评论的方法。在 2002 年的 ACL 大会上和 2003 年的 EMNLP 大会上，情感分析这一概念再次被多为学者作为论文的主要论点进行发表[24][25]。之后几年意见挖掘、情感分析、倾向分析成为了文本分析领域和自然语言处理领域的热点研究课题，并成为文本分析领域的主流研究方向。

在国内，文本倾向分析也获得了学术界的关注，各种以文本倾向性分析为主要主题的计算机会议也纷纷在国内举办，其中比较有代表性的会议有 2008 年，由 CIPSC 主办的中文倾向性分析评测（Chinese Opinion Analysis Evaluation, COAE2008）会议[26]，该会议提出了关于中文倾向性分析的主要任务，阐述了中文倾向性分析的难点和要点。由于中文与英文的不同，没有固定的分割符区分单词，而且中文存在歧义导致中文分词与

英文分词的难度不在一个级别，而文本倾向分析又依赖于文本切分和处理的精确性，所以中文的倾向性分析相较于英文来说更加具有挑战性。

可以发现，2001 年以后的近几年是文本分析领域发展的重要时期，这期间在各大顶级会议期刊上发表了近百篇有关情感分析和意见挖掘的论文，同时，文本倾向分析的研究也逐渐被应用到生产环境中为人们的生活带来便利，产生这一现象的主要原因包括机器学习领域在自然语言处理中的应用，以及互联网的发展产生了大量的可用来研究的语料数据，同时，人们对软件服务质量的需求提升也促进了该领域的发展。

但是基于机器学习的文本分析算法需要训练标注好的文本获得训练模型，这种方式对文本的特征提取方式依赖很大，随后有些学者提出了基于神经网络算法的方式进行文本分析[27][28][29]，而且训练的文本模型的准确度更高。

1.2.2 基于大数据平台的文本分类综述

上一节提到了文本倾向分析领域的发展，其中涉及到如何高效的对文本特征进行分类的问题，随着互联网技术越来越成熟，互联网与人们的生活息息相关，但是随之而来的是以指数级速度增长的信息数据，普通的计算节点已经无法承受如此规模的计算任务，2003 年 Google 公司发表了 MapReduce[2]计算模型，该论文阐述了如何使用廉价的计算节点构造计算集群来处理大数据计算任务，2005 年 Apache 基金会开源了 Hadoop 分布式框架，该框架提供了一套进行 MapReduce 计算的分布式系统，这使得各个研究机构和公司可以应用大数据计算来处理相关的课题和业务，同时大数据也成为研究热点进入人们的视线。

2009 年 Spark 项目在加州大学伯克利分校 AMPLab 成立，并在 2014 年成为 Apache 的顶级项目，其出色的运行效率以及丰富的操作迅速成为大数据计算中的佼佼者，并被应用到各个领域，在文本分类领域，大数据平台也被应用到其中作为提高训练速度的工具，同时，企业和用户对文本处理任务的需求越来越多，也促进了以文本处理为核心的大数据平台的推出，比较具有代表性的是 2016 年 Google 推出的 Cloud Natural Language API，以及 2016 年微软推出的 Natural Language Processing Tools，这两个工具包都是针对 NLP 领域的并行计算工具，其中容纳了许多自然语言处理相关的算法，包括文本表示、特征提取、文本分类等相关算法。

但是，这两个工具包并没有提供针对中文文本分析接口，本文采用 Spark 大数据平台实现了针对中文文本的倾向性分析系统，利用 spark 的分布式计算能力，完成中文文本的切分、过滤、特征提取以及模型训练等过程。

1.3 本文的主要工作

1.3.1 研究内容

(1).基于神经网络的文本特征提取算法

目前常用的语言模式是基于概率统计的语言模型，该类语言模型都是基于统计学知识数值化表示自然语言并对其建模，但是这种语言模型忽略了词与词之间的搭配关系，在训练过程中，会丢失部分特征。本文使用神经网络语言模型对文本建模，并融合统计语言模型中的优点，既保留了文本的语义特征，又参考了文本的统计特性，设计一种融合这两种模型的文本特征提取算法。

(2).文本分析算法的并行化技术

文本分析需要进行模型训练，训练过程与计算过程息息相关，当训练语料达到一定量级后，原有算法的效率会越来越低，这时就需要改进算法的计算过程，将可以并行计算的地方改成并行化执行。本文通过研究文本分析算法的并行化技术，针对文本特征提取算法，设计了基于 Spark 平台的并行化模型。通过并行化模型，改进算法的计算过程，将可以并行计算的步骤分布到 Spark 集群上，提高了整个算法的计算效率，节省了计算时间成本。

(3).中文文本分析流程

中文文本分析流程包括语料摄取、语料标注、语料存储、模型训练和模型验证，其中语料摄取是指如何获取训练文本，常用的方法是使用爬虫摄取所需语料，但是，一般的爬虫不能过滤摄取的文本，本系统采用定制化爬虫精确摄取语料，提高初始语料质量。语料标注是对文本进行正负向标注，标注质量决定了训练模型的准确度，本文采用标签和情感词典融合的方法对语料进行标注，提高标注效果。语料存储需要提供高效的读写语料的能力，为模型训练提供基础服务。模型训练和模型验证是文本分析的重要环节，本文采用 Spark 平台作模型训练的主要平台。

(4).中文文本分析原型系统的设计与实现

根据中文文本分析流程，设计基于 Spark 平台的文本分析系统，系统由文本预处理、文本存储、文本分析三部分组成，其中文本预处理部分负责语料摄取、语料标注，文本存储部分负责语料库建立和维护，文本分析部分负责文本模型训练、模型验证。通过对原型系统的搭建，完成文本特征提取算法的实现、文本分析算法的并行化，并给出系统验证、测试和分析结果。

1.3.2 创新之处

中文文本倾向分析是自然语言处理中非常重要的研究课题，随着技术的不断进步，对文本的分析算法也在不断的更新换代，最初的文本倾向分析是基于规则和情感词典的

文本结构分析，这个阶段的分析算法需要专家配置大量的语法规则来分析文本倾向，随着机器学习算法的发展，研究者们将机器学习算法应用到自然语言处理领域，并且在分词、特征提取等方面获得不错的效果。进入 21 世纪，信息技术发展迅速，大数据时代到来，同时，随着神经网络技术在语音图像等领域的广泛应用，也激发了研究者使用神经网络算法处理 NLP 问题，本课题基于这些学者的研究针对文本倾向分析领域提出主要创新点有：

(1).设计基于 LDA 和 Doc2vec 算法的文本特征提取算法，文本的特征提取是作为文本分类任务的前提条件，以往的文本分类任务采用传统的文本特征表示方式，本课题采用最新的神经网络模型 Doc2vec 并结合主题模型 LDA 来作为文本的主要特征进行模型训练，提高文本分类的准确率。

(2).针对文本特征提取算法的并行化实现，文本特征提取任务是密集型计算任务，并且随着语料的不多增加，训练时长也会相应变长，本课题将文本特征提取算法移植到 Spark 平台，并实现算法的并行化来提高训练速度，大大的节约了训练成本。

(3).基于 Spark 大数据平台实现中文分词的并行化，由于本课题是针对中文文本进行分析，所以需要对初始文本进行分词处理，以往的分词程序都是在单机环境下运行，如果语料十分庞大，会影响分词效率，本课题使用 Spark 大数据平台对分词进行并行化处理提高分词的速度。

(4).提出了基于标签和情感词典融合的语料标注策略，语料标注是训练模型的前提，只有准确的标注语料，才能训练出准确的模型，基于人工标注的方式费时费力，而完全自动化的标注准确率不高，本课题提出了融合标签和情感词典的标注策略来提高标注准确度。

1.4 本文的组织结构

根据以上阐述的内容，本文的组织结构安排如下：

第一章 绪论部分。本章主要对文本分析理论的研究和大数据平台背景进行阐述，并分析了国内外在相关领域的研究成果和本课题的创新之处，最后结合研究背景归纳了本文的主要研究内容。

第二章 中文文本分析算法及关键技术。本章主要是对中文文本分析算法和关键技术的介绍，详细的介绍了和本课题相关的文本分析算法的细节，通过本章节的介绍可以为后面的章节打下理论基础。

第三章 应用大数据平台处理文本数据。本章主要介绍 Spark 大数据平台的特点及内部原理，通过本章的介绍，阐述了为什么要使用大数据平台处理文本数据，以及如何使用大数据平台处理文本数据来提高计算效率。

第四章 基于电影评论的文本倾向分析系统的设计。本章介绍了基于电影评论的中文文本倾向分析系统的总体设计思路，以及总体框架设计。通过对文本倾向分析系统进行总体框架设计，倾向分析算法和特征提取算法的并行化实现三个小结来阐述整个系统的设计过程。

第五章 基于电影评论的文本倾向系统实现及验证测试。本章节给出了整个系统的实现过程，通过对各个模块的分析来阐述系统的实现过程。其中包括文本预处理模块的实现，文本存储模块的实现，文本分析模块的实现。并对系统进行验证，给出结果分析。

第六章 结束语。本章是对整个课题的总结，并给出对研究课题的进一步展望。

第二章 中文文本分析算法及关键技术

2.1 文本分析相关知识点介绍

文本分析中涉及了许多自然语言处理的专业术语，这些术语是研究文本的基础，本小结通过三方面来讲解这些关键知识点，其中包括语言模型及文本表示方法、文本特征及特征抽取方法、文本分类及分类模型介绍。这三个方面是进行文本倾向分析过程中必须经历的三个关键过程，通过对文本的一系列处理，最终可以得到训练后的模型，之后可以通过系统提供的接口，根据具体的需求对训练模型进行应用，完成实际的任务需求。

2.1.1 语言模型及文本表示方法

(1). 语言模型

语言模型[30]是用来对自然语言进行建模的方式，传统的语言模型是统计语言模型 (Statistical Language Model)，它是一个表示语言片段的概率分布函数，其数学表达形式如下：

$$p(W) = p(w_1^T) = p(w_1, w_2, \dots, w_T) = \prod_{t=1}^T p(w_t | \text{Context}) \quad (1)$$

其中 $W=w_1^T=(w_1, w_2, \dots, w_T)$ 表示由 T 个词 w_1, w_2, \dots, w_T 按顺序构成的语言片段，则 $p(W)$ 代表这些词组合在一起的概率。根据贝叶斯公式，可以将 $p(W)$ 链式的分解为：

$$p(w_1^T) = p(w_1) \cdot p(w_2 | w_1) \cdot p(w_3 | w_1^2) \cdots p(w_T | w_1^{T-1})$$

统一将每个词的上下文记作 Context ，则可得到(1)式的表达形式。根据对 Context 的划分策略不同，可以形成不同的语言模型，常见的有 $n\text{-gram}$ 模型、 $n\text{-pos}$ 模型、决策树模型、最大熵模型、最大熵马尔科夫模型、神经网络语言模型[30]等方法，对于文本的不同建模方式会有不同的效果，同时各语言模型也各有特点，以下是对各语言模型的介绍：

● $N\text{-gram}$ 模型

$n\text{-gram}$ 模型认为某个词出现的概率与其前面 n 个词有关，最极端的情况是 $n=1$ 的时候，即某个词出现的概率只与该词本身有关。这种语言模型称为上下文无关模型，即：

$$p(w_t | \text{Context}) = p(w_t) = \frac{N_{w_t}}{N}$$

当 $n \geq 2$ 时是称为上下文相关模型，在一般应用中取 $n=2$ 或 $n=3$ ，即 Bigram 或 Trigram 。 $n\text{-gram}$ 模型的优点是考虑了前 $n-1$ 个词的因数，而这 $n-1$ 个词在自然语义上有很强意义，同时采用 Bigram 或 Trigram 的方式可以大大简化计算规模，提高效率。但是 $n\text{-gram}$

语言模型自身也有一定的局限性，例如：**n-gram** 模型依据语料的关系，语料不足训练的结果一般不理想，而且这种模型忽略了词之间的相似关系，只考虑了词与上下文的关系而没有考虑词与词之间的关系，其次 **n-gram** 模型在有些元组没有出现过的情况下会出现统计概率为 0 的情况，这会导致整个语言序列的概率为 0，出现这种情况往往需要进行修正才能获得准确的结果。

● N-pos 模型

n-pos 模型是基于 **n-gram** 模型的基础上衍生的一种语言模型，**n-pos** 模型是基于这样的假设：单单考虑前 n 个词不足以表示当前词的特征，而自然语言中的词语搭配往往是根据词的语法功能决定的，所以 **n-pos** 将当前词的前 n 个词按照语法功能进行分类，由这些词来决定当前词的概率，该分类叫做 **Part-Of-Speech**，即 **n-pos** 算法的由来，**n-pos** 模型的条件概率公式如下：

$$p(W) = p(w_1^T) = p(w_1, w_2, \dots, w_T) = \prod_{t=1}^T p(w_t | c(w_{t-n+1}), c(w_{t-n+2}), \dots, c(w_{t-1}))$$

c 是词性映射函数， $c(w_t)$ 表示将单词 w_t 映射为其所属的词性类别，如果一段语言序列有 T 个词， K 种词性分类，则可以将 **n-gram** 的条件概率求解过程从 T^{n-1} 可能变成 K^{n-1} 种，大大的提高了计算效率，而且这种效率的提高也不会影响准确率的下降。

● 决策树模型

决策树模型是为了解决 **n-gram** 模型中对相似的词类搭配重复求解导致的效率损失问题，构造一颗决策树表示当前词的上下文，由对根节点提问的不同回答进入子节点，直至叶子节点，从而得到当前词上下文的分布信息。为构造决策树，需要预先定义一个关于上下文信息的问题集和一个评论问题的优劣函数。

决策树模型的优点是该模型是一种通用的语言模型，**n-gram** 模型、**n-pos** 模型都可以用决策树模型表示出来，而且决策树中的分布信息不是固定好的而是根据训练语料库得到的，但是，由于决策树模型的抽象性导致这种模型的构造难度偏大而且时间复杂度和空间复杂度都很高。

● 最大熵模型

最大熵模型的基本思想是：对一个随机事件的概率分布进行预测时，若概率模型需要满足一些约束，则在满足约束的情况下对未知的情况不做任何假设，这种情况下，概率分布最均匀，得到的概率分布的熵也最大。最大熵语言模型的概率分布公式如下所示：

$$p(w_t | Context) = \frac{e^{\sum_i \lambda_i f_i(context, w)}}{Z(context)}$$

其中 λ_i 是参数， $Z(context)$ 为归一化因子。

● 神经网络语言模型

神经网络语言模型(Neural Network Language Model)是近几年比较流行的语言模型，即通过神经网络算法来拟合语言模型的方式求概率分布的最大似然估计，NNLM 采用的

是 Distributed Representation 来表示词向量, 神经网络语言模型与 n-gram、n-pos、决策树模型、最大熵模型最大的区别是前者是基于统计的语言模型, 后者是基于深度学习的语言模型, 这些模型所关注的结果都一样, 但是建模思路不尽相同。

常用的 NNLM 模型有 C&W 的 SENNA、M&H 的 HLBL、Mikolov 的 RNNLM[31][32][33], 其中比较经典的是 Bengio 等人在 2001 年发表在 NIPS 上的文章《A Neural Probabilistic Language Model》[14], 该文章用一个三层的神经网络模型来构建语言模型, 这三层网络分别是输入层, 即以词向量首尾相连得到一个很长的向量作为输入向量, 第二层是隐藏层, 使用 $d+Hx$ 计算所得, d 是一个偏执项, 然后使用 \tanh 作为激活函数, 最后一层是输出层, 共有 $|V|$ 个节点, 其中 $|V|$ 表示词表的大小, 每个节点 y_i 表示下一个词为 i 未归一化 \log 概率。最后使用 softmax 激活函数将输出值 y 归一化成概率。整个模型的计算公式为:

$$y = b + Wx + U \tanh(d + Hx)$$

式子中的 U 是隐藏层到输出层的参数, 整个模型多数的计算集中在 U 和隐藏层的矩阵乘法中。式子中还有一个矩阵 W , 这个矩阵中的元素表示输入层和输出层的边关系。这些边关系就是从输入层直接到输出层的一个线性变换, 如果不需要直连边的话, 将 W 置为 0 就可以了。

(2). 文本表示方法

$$\begin{matrix} doc_1 \\ doc_2 \\ doc_3 \end{matrix} \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

文本表示方法是指将文本向量化, 转换成可以计算的数学矩阵, 一般是将文档作为矩阵中的行(如上图), 单词作为矩阵中的列, 而元素就是需要数字化的表示方式, 在机器学习领域常用的表示方法是 Bag-Of-Word(BOW)模型和 Bag-Of-N-gram 模型, 而在深度学习中常用的模型是 Word Representation 或 Word Embedding, 后者通常叫做词向量模型。以下是对文本表示方法的介绍:

● 词袋模型

词袋模型是进行文本分类中经常用到的文本表示方式, 因为这种方式方便快捷, 易于理解, 但是同时词袋模型忽略了词语之间的顺序, 所以词袋模型一般应用在那些对词序要求不高的文本分析领域, 词袋模型通常需要一个词典保存文档集合中所有出现过的单词, 并对单词进行编号, 然后文档可以表示成一个一维向量, 其中向量维度为词典的大小, 向量元素一般单词的出现的频率(也可以用 TF-IDF 表示), 以下通过例子来讲解词袋模型的原理。

假设有两个文档, 文档 1 为“我 爱 北京 天安门, 我 是 中国人”, 文档 2 为“她 喜欢 看 电影, 她 是 美国人。”则根据文档 1 和文档 2 可以构造一个词典: {1:“我”, 2:“爱”, 3:“北京”, 4:“天安门”, 5:“是”, 6:“中国人”, 7:“她”, 8:“喜欢”, 9:

“看”，10：“电影”，11：“美国人”}，根据词典大小可以将文档表示成为一个 11 维的向量，其中每个元素是单词在词典中出现的频率。

文档 1: [2,1,1,1,1,1,0,0,0,0,0]

文档 2: [0,0,0,0,2,0,1,1,1,1,1]

为了提高词袋模型的实用价值可以将文档中的元素用单词的 TF-IDF 值表示，这样可以提高单词和文档的特征关系。

● Ngram 词袋模型

以上提到的词袋模型是最简单的文本表示方式，其没有考虑单词与单词之间的搭配关系，这种表示方式丢失了很多文本特征，在文本分类过程中使用这样的模型得到的准确率往往不高，Ngram 词袋模型是对这种模型的优化，即使用 n-gram 语言模型得到单词的条件概率，用该条件概率表示单词和文档，这种表示方式保留了单词之间的搭配关系，但是无法获得单词和文档之间的关系，即无法获得语义相近的单词，丢失了文档与文档的特征，模型不够平滑。为了使得模型更加平滑，还需要采用平滑处理来优化模型。

● 词向量模型

词向量是指将自然语言中的词用数学化的向量表示，这样做的好处是可以对词进行数学公式计算，把计算机无法理解的文字数字化，常见的对词向量的划分有 one-hot representation，就是用一个只有一项为 1 其他项全为 0 的长向量表示一个词，向量的维度 N 为词典的大小，向量中为 1 的项对应该词在词典中的索引，例如对于“中国”这个词，假设其在字典中的索引为 2，这“中国”可以表示成如下的词向量：

$$[0, 1, 0, \dots, 0]^T$$

这种词向量表示方法虽然简单，但是其只标识了单个词的信息，向量十分稀疏，而且向量的维度很高，这样会导致维数灾难，不适合作为计算机运算的输入，这种表示方式还有一个问题是向量中没有词与词之间的关系，即词的上下文关系，所以在训练过程中会丢失许多特征信息，其数学意义不大，所以很少被使用。

另一种对词向量的表示是 Distributed Representation，其基本的思想是：通过训练将自然语言中的词映射成为一个长度固定的向量，该向量维度一般比较短，这样所有的词向量可以构成一个向量空间，同时由于向量长度较短，也有利于对其进行相应的数学运算从而研究它们之间的关系。这种词向量表示方式比较符合计算机输入参数的要求，且自带平滑，保留了词的上下文特征和词与词之间的特征，是比较理想的文本表示模型。

2.1.2 文本特征及特征抽取

文本特征[30]是表示文本的基本单位，文本的特征抽取是文本挖掘领域中重要的研究课题，对于文本来说其特征一般都具备一定的特性。首先，文本特征项需要能够标识文本的主要内容。其次，特征项需要代表文本的特征属性，即与其他文本的特征有明显

区别。最后，特征项的维度不宜过多，容易导致维度灾难。在中文领域，中文文本由字作为基本单元组成，但是字所包含的特征不足够表示文本的特征，相对于单个字来说，词和短语所包含的信息更加丰富，所以一般采用词或短语作为文本的基本特征单元。如果用词或短语作为文本的特征项，随着文档的规模越来越大，特征向量包含的特征值就会非常巨大，这样会导致维度灾难，无法进行计算。这时就需要借助特征抽取技术来完成文档的特征抽取(或文档的稀疏表示)，特征抽取需要完成的任务是在不丢失文本核心信息的情况下，用更少的特征表示文本，即将原来稠密矩阵用稀疏矩阵进行表示。

特征抽取的方式一般有 3 种[34]：(1) 通过映射或变换的方法减少特征的维度，(2) 通过排序获取最具代表性特征值，(3) 用机器学习的方法训练文本获取特征模型。以下是基于统计的特征提取方法：

(1) TF-IDF:

TF-IDF(term frequency-inverse document frequency)算法是用来衡量单词权重的统计方式，其中 TF 指为单词频率，IDF 为逆文档频率，即该词出现在文档中频率的倒数，该参数用于计算该词区分文档的能力。TF-IDF 之所以能作为衡量单词权重的特征是因为其基于这样的思想：如果单词在一篇文档中出现的频率很高而在其他文档里出现的概率很低，那么说明这个单词对这篇文档很重要，也就是该单词能作为这篇文档的特征。TF-IDF 通常用来提取文档的关键词特征，即通过排序获取 TF-IDF 值高的单词作为文档的关键词来代表文档的特征。

(2) 互信息 (Mutual Information)

互信息(Mutual Information)，顾名思义，是指两个随机变量间相互依赖的程度，该定义是概率论和信息论中的重要概念，其与相关系数的区别在于它不仅仅局限于实值性的随机变量，其定义更加一般，可以应用到自然语言处理中。互信息的定义如下：

$$I(X, Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

其中 X, Y 为随机变量，其联合分布为 $p(x, y)$ ，边际分布为 $p(x), p(y)$ ，互信息 $I(X, Y)$ 是联合分布 $p(x, y)$ 与乘积分布 $p(x)p(y)$ 的相对熵。互信息可以作为抽取文本特征词的度量单位，通过设定单词与某一类别的互信息阈值来抽取文本的特征词。

(3) 信息增益方法 (Information Gain)

信息增益 Information Gain (也叫 information divergence, relative entropy)，是信息论中一种度量单位或距离，其于熵的定义联系紧密。信息增益定义为一个特征可以为系统带来多少信息，通过增加某一特征或去掉某一特征，系统信息熵的变化来定义信息增益。其可用来抽取分类系统中的特征项，如果某个特征项的信息增益超过某一阈值，则可以用该特征项作为系统的特征值。

(4) 主成分分析法 (Principal Component Analysis, PCA)

主成分分析是通过降维[35]的方式提取数据集的特征，在降维后，保留对数据集方差影响最大的特征。主成分分析通常需要对特征矩阵进行矩阵分解来获取特征向量，对于文本分析来说，特征矩阵的维度不定，所以其特征矩阵不是严格意义上的 $N \times N$ 矩阵，需要用到 svd 算法对其进行分解获取特征向量，主成分分析的优点是可以对维度很大的特征进行降维表示，不仅可以降低运算成本，还同时保留了系统的特征。

(5) n -gram 算法

n -gram 算法的思路比较简单， n -gram 算法将文本片段看作连续的输入流，同时设置一个大小为 N 的滑动窗口，通常情况下 N 的取值不会很大，从输入流开始使用滑动窗口对文本进行切分，形成大小为 N 的文本片段，然后对这些文本片段进行统计，获取文本的特征文本片段，这既是文本的特征向量， n -gram 算法可以应用到那些很难获取特征项的文本中，中文没有固定的分割符，可以利用 n -gram 算法作为中文分词的一种算法。

特征提取是为了获取文本的特征，需要根据特定场景需求选择特定的特征提取方式，对特征的提取不局限于单一的方法，往往需要通过多种方法对基础语料进行处理才能获得比较理想的特征项[36]，本课题采用特征抽取的第三个方法，即通过机器学习方法训练文本获取特征模型，本文采用 LDA 模型错误!未找到引用源。作为文本特征提取算法，LDA 算法是以 N -gram 算法为基础的文本表示算法，其可以将文本进行稀疏表示获得文档与主题模型，该模型可以作为文档特征进行文档分类任务。

2.2 基于主题模型和文本向量的文本倾向分析算法

本节主要介绍本课题使用的文本倾向分析算法，通过上一个章节的介绍，对文本分析领域的基本概念有一定的了解，文本分析需要对文本建模，并提取特征，之后需要对文本进行训练获取模型，然后通过文本分类算法对文件进行分类，获取文本的正负倾向性。本文采用 LDA 模型[17]获取文本主题作为文本特征的一部分，然后结合神经网络模型训练获取词向量，并训练得到 word2vec 模型[32]，最后通过随机梯度下降算法对文本进行分类，以下是对这三个算法的原理介绍。

2.2.1 基于 LDA 的文本特征提取算法

隐含狄利克雷分布 (Latent Dirichlet Allocation, 简称 LDA) [17] 是一种主题模型，该模型可以将文档中的主题以概率分布的形式给出，从而可以通过分析文档抽取出文档的主题，我们利用 LDA 算法的这一特性可以作为文本特征抽取的一种方法，因为对于文档来说，其主题就是该文档的特征，同时，LDA 算法是基于 N -gram 词典模型的，即其认同一篇文档是有一组词构成，词与词之间没有先后关系。LDA 的主要思想[37][38]是：

给定一篇文档，该文档的生成方式如下：

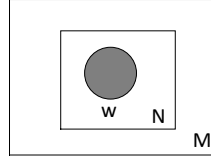
- 从 Dirichlet 分布 α 中取样生成文档 i 的主题分布 θ_i
- 从主题分布 θ_i 中取样生成文档 i 第 j 个词的主题 $z_{i,j}$
- 从 Dirichlet 分布 β 中取样生成主题 $z_{i,j}$ 对应的词语分布 $\phi_{z_{i,j}}$
- 从词语的多项式分布 $\phi_{z_{i,j}}$ 中采样最终生成词语 $w_{i,j}$

其中，Dirichlet 分布[39]是多项式分布的共轭先验概率分布。

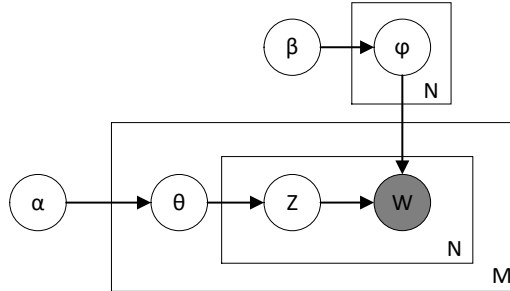
假设有一篇文档 W ，其由 N 个词组成，记作 $W=(w_1, w_2, w_3, \dots, w_N)$ 。用 $p(w_n)$ 表示词 w_n 的先验概率，则生成文档 W 的概率为：

$$p(W) = \prod_{n=1}^N p(w_n)$$

其图模型为：



图中灰色部分 w 表示可观测变量， N 表示一篇文档中共包含 N 个单词， M 表示有 M 篇文档。则可以得到 LDA 算法的概率模型图如下所示：



其中，其中 ϕ 表示词分布， θ 表示主题分布， α 是主题分布 θ 的 Dirichlet 分布的参数， β 是词分布 ϕ 的 Dirichlet 分布的参数， N 表示单词总数， M 表示文档总数。

LDA 算法根据先验知识 α 确定文档的主题分布 θ ，然后从该主题分布中抽取主题 Z ，然后根据先验知识 β 获得在主题 Z 下的词语分布 ϕ ，最后从主题 Z 对应的词语分布中抽取单词 w ，重复以上过程 N 词，就生成了文档 W 。 α, β 是 Dirichlet 分布的两个参数，即 LDA 算法通过 Dirichlet 分布来估计文档的主题分布和词分布，获取文档的概率。

由上所述，我们知道基于 N-gram 词袋模型的文本是服从多项式分布的，而 Dirichlet 分布正好是多项式分布的先验分布，同时 LDA 模型是从 pLSA 主题模型思想上衍生出的模型，所以在深入了解 LDA 模型之前我们需要了解 pLSA 模型[40][41]和 Dirichlet 分布。

1). 概率潜在语义分析(probabilistic latent semantic analysis, 简称 pLSA)

pLSA是一种文本主题模型,文本主题模型的主要思想是:一篇文章由多个主题构成,多个主题出现的概率不同,每个主题中包含多个单词,每个单词出现的概率也不同,则生成一篇文章的过程是首先选择一个主题,然后从该主题选择一个词,重复执行 N,则可以生成一篇包含 N 个词的文章。上述过程抽象后就是 PLSA 模型,则可得到 PLSA 模型描述如下:

定义:

$P(d_i)$ 表示从海量的文档中选取某篇文档的概率。

$P(w_j|d_i)$ 表示在给定文档 d_i 的情况下出现词 w_j 的概率。

$P(w_j|d_i)$ 是一个可计算的概率,对于每个词语, $P(w_j|d_i)$ 表示该词语在该文档出现的频率除以文档中词语的总数。

$P(z_k|d_i)$ 表示在给定文档 d_i 的情况下包含主题 z_k 的概率。

$P(w_j|z_k)$ 表示在给定主题 z_k 的条件下出现某个词 w_j 的概率。

则根据以上定义可以得到生成一篇文档的步骤如下:

以概率 $P(d_i)$ 选择一篇文档 d_i

以概率 $P(z_k|d_i)$ 从主题分布中选择一个隐含的主题类别 z_k 。

以概率 $P(w_j|z_k)$ 从词分布中选择一个单词 w_j 。

LDA 模型是在 PLAS 模型的基础上增加了两个 Dirichlet 先验估计,参数估计是文本分析常用的方法[42][43], LDA 模型生成文档的方式是:

以概率 $P(d_i)$ 选择一篇文档 d_i 。

从 Dirichlet 分布中以 α 为参数取样生成该文档的主题分布 θ_i 。

从主题分布 θ_i 中选择文档 d_i 第 j 个词的主题类别 $z_{i,j}$ 。

从 Dirichlet 分布中以 β 为参数取样生成主题为 $z_{i,j}$ 对应的词语分布 $\phi_{z_{i,j}}$ 。

从词分布 $\phi_{z_{i,j}}$ 中选择一个单词 $w_{i,j}$ 。

接下来,我们就需要知道为什么选择 Dirichlet 分布作为主题分布和词语分布的先验概率分布。

2).Dirichlet 分布

在正式了解 Dirichlet 分布之前,首先我们从最简单的二项分布入手,二项分布即重复 n 次独立的伯努利试验。当试验次数为 1 时,二项分布服从 0-1 分布。二项分布的概率密度函数为:

$$P(K = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

其中, 对于 $k \in \mathbb{R}$, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ 是二项式系数, 记为 $C(n, k)$ 。

把二项分布推广到多个互斥事件就得到了多项分布, 多项分布的概率密度函数为:

$$P(x_1, x_2, \dots, x_k; n, p_1, p_2, \dots, p_k) = \frac{n!}{x_1! \dots x_k!} p_1^{x_1} \dots p_k^{x_k}$$

而 Dirichlet 分布的概率密度为:

$$f(x_1, x_2, \dots, x_k; \alpha_1, \alpha_2, \dots, \alpha_k) = \frac{1}{B(\alpha)} \prod_{i=1}^k x_i^{\alpha_i-1}$$

其中:

$$B(\alpha) = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^k \alpha_i)}, \sum x_i = 1, \Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \quad (\text{即 Gamma 函数}).$$

由证明可得(证明略):

$$\text{Dir}(\vec{p}|\vec{\alpha}) + \text{MultCount}(\vec{m}) = \text{Dir}(\vec{p}|\vec{\alpha} + \vec{m})$$

其中 Dir 表示 Dirichlet 分布, MultCount 表示多项分布, 即上式为 Dirichlet-Multinomial 共轭, 也就是说 Dirichlet 分布和 MultCount 分布为共轭关系。由共轭先验的定义可知, Dirichlet 分布是 MultCount 的共轭先验分布, 同时, 基于 N-gram 的文本主题模型其实质是一个多项式分布, 所以 LDA 模型采用 Dirichlet 作为主题模型的先验概率分布, 来估计主题参数。

综上所述, 使用 LDA 模型可以获取文章的主题分布, 而文章主题分布是标识文章特征的更加高级的特征项, 我们可以使用 LDA 模型提取出文本的主题作为特征值, 这些特征值可以作为文本分类任务的参数, 完成有关文本分析的需求。

2.2.2 基于 word2vec 的文本表示算法

上一节中提到了 NNLM 模型, word2vec 是基于 NNLM 模型训练语言模型的算法, 其是被 Mikolov 在 2013 年提出[31], 并相继完善[32][33]。Word2vec 算法是通过神经网络语言模型拟合自然语言概率模型, 从而获得计算语言概率模型的高效方法, 在训练过程中, word2vec 会获得语料库中每个词的词向量, 该词向量是 Distributed Representation 类型的词向量, 其维度由输入参数决定, 一般在 100-500 维。根据词向量我们可以得到一个词向量空间, 该词向量空间包含了词本身的特征以及词与词之间的特征。

本节主要阐述 word2vec 的算法原理[44][45], word2vec 中主要运用了 CBOW 模型 (Continuous Bag-of-Words Model) 作为其语言模型, 其中 CBOW 模型是在已知当前词 w_t 的上下文 $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ 的前提下预测当前词 w_t 的概率。由上一节可知自然语言的语言模型为 (1) 式所示, 而基于神经网络的语言模型通常取其对数似然函数作为目标函数:

$$L = \sum_{w_t \in C} \log p(w_t | \text{Context}(w_t)) \quad (2)$$

word2vec 构造了一个三层的神经网络：输入层、投影层和输出层，这三层的含义分别是：

(1).输入层：包含 $\text{Context}(w_t)$ 中 $2c$ 个词的词向量，即：

$$v(w_{t-c}), \dots, v(w_{t-2}), v(w_{t-1}), v(w_{t+1}), v(w_{t+2}), \dots, v(w_{t+c}).$$

(2).投影层：将输入层的向量做求和累加，即：

$$X_w = \sum_{i=1}^{2c} v(w_i), (w_i \text{ 为 } w_t \text{ 的上下文单词})$$

(3).输出层：输出层对应一棵 Huffman 树，Huffman 树是以语料中各词的词频为权值构造而成的，其叶子结点为语料中的词。

构造了以上神经网络模型后，主要的问题是解决如何通过以上神经网络模型来拟合条件概率 $p(w_t | \text{context}(w_t))$ ，word2vec 将其视作一个二分类问题，利用逻辑回归的方法求每个结点的概率，即从 Huffman 的根结点到叶子结点的过程，将左结点分为负类，右结点分为正类，利用逻辑回归，易知，一个结点被分为正类的概率是：

$$\sigma(X_w^T \theta) = \frac{1}{1 + e^{-X_w^T \theta}} \quad (3)$$

显然，其被分为负类的概率就是： $1 - \sigma(X_w^T \theta)$

其中， $\sigma(x) = \frac{1}{1+e^{-x}}$ 为 sigmoid 函数，其作为神经网络的激活函数， X_w^T 为词向量， θ 为参数向量，其代表 Huffman 树中的非叶子结点向量。

根据以上定义就可以得到计算 $p(w_t | \text{context}(w_t))$ 的算法：对于词典 D 中的任意词 w ，Huffman 树中必定存在一条从根结点到词 w 的路径 p^w 。路径 p^w 上存在 $l^w - 1$ 个分支，将每个分支看做一次二分类，每一次分类就产生一个概率，将这些概率乘起来，就是所需的 $p(w_t | \text{context}(w_t))$ 。

于是，可以得到条件概率 $p(w_t | \text{context}(w_t))$ 的公式为：

$$p(w_t | \text{context}(w_t)) = \prod_{j=2}^{l^w} p(d_j^w | x_w, \theta_{j-1}^w) \quad (4)$$

其中：

$$p(d_j^w | x_w, \theta_{j-1}^w) = [\sigma(x_w^T \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(x_w^T \theta_{j-1}^w)]^{d_j^w} \quad d_j^w = 0 \text{ or } 1 \quad (5)$$

表达式 $p(d_j^w | x_w, \theta_{j-1}^w)$ 中， d_j^w 表示路径 p^w 中第 j 个结点对应的 Huffman 编码， θ_{j-1}^w 表示路径 p^w 中第 $j-1$ 个非叶子结点对应的向量。

将上式带入对数似然函数(2)，可得：

$$L = \sum_{w \in C} \log \prod_{j=2}^{l^w} [\sigma(x_w^T \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(x_w^T \theta_{j-1}^w)]^{d_j^w}$$

$$= \sum_{w \in c} \sum_{j=2}^{l^w} \{(1 - d_j^w) \log[\sigma(x_w^T \theta_{j-1}^w)] + d_j^w \log[1 - \sigma(x_w^T \theta_{j-1}^w)]\} \quad (6)$$

其中，记 $L(w, j)$ 为：

$$L(w, j) = \{(1 - d_j^w) \log[\sigma(x_w^T \theta_{j-1}^w)] + d_j^w \log[1 - \sigma(x_w^T \theta_{j-1}^w)]\} \quad (7)$$

则上式即为 CBOW 模型的目标函数，使得该函数最大化，当超过某个阈值时，就可以用来判断语言模型是否是自然语言，Word2vec 通过采用随机梯度上升法来计算该目标函数。

基于神经网络语言模型的训练，其是无监督的训练算法，不需要进行人工识别，只需要将文本切分成单个词语，将切分后的语料进行训练就可以得到语言概率模型，大大节约了训练成本，而且神经网络语言模型通过滑动窗口的模式保留了词语上下文的关系，这样对语义分析也有很重要的意义。

同时，对于 word2vec 神经网络语言模型中涉及的参数，都是由样本训练得到的，这有别于通常的机器学习算法，需要在训练样本前给定参数，所以词向量和概率模型都是在训练中得出，词向量作为训练过程中的副产品对解决自然语言领域中的问题有极大的参考作用，同时，其无监督的训练方式和高效的训练速度，也是 word2vec 算法受到人们青睐的重要原因。

2.3 文本分析的关键技术

文本分析是一个综合学科，其覆盖了许多知识点，除了前几节提到的有关算法方面的内容，还包含文本采集技术、文本切分技术、文本标注技术等，这些技术是进行文本分析的基础，掌握这些关键技术，在处理复杂的文本分析任务时会游刃有余，不仅可以提高效率而且还可以灵活的应对不同的需求，本节通过三个方面来对文本采集技术，文本切分技术和文本标注技术进行阐述。

2.3.1 语料采集关键技术

语料采集通常是采用爬取技术爬取网络上的数据进行分析，或者是收集自有的日志数据、数据库数据进行分析，网络爬虫的优点是可以根据需求收集各种类型的文本数据训练模型，缺点是网络上的文本数据大多是没有进行标注的文本，而且存在大量的网络用语，需要进行清洗才能满足训练要求。

网络爬虫一般分为通用爬虫和定制爬虫两种类型，通用爬虫一般是为搜索引擎等系统设计的爬虫，其目的是快速的爬取互联网上的内容，而定制爬虫一般是针对某一特定领域或网站进行爬取，目的是获取特定的资源，在效率方面定制爬虫不如通用爬虫那样

高效，但是在质量上，定制爬虫比通用爬虫更加出色。

作为语料采集任务，一般采用的是定制爬虫来采集文本信息，在效率容忍的范围内可以获取到符合需求的训练语料，定制爬虫一般都具有对特定的网页内容进行过滤的能力，这样可以精确的获取的想要的文本内容，达到初次过滤文本的目的，例如对于网站的评论信息，定制爬虫可以将评论主题，评论时间等有关评论的文本爬取下来，并对每个字段分别保存，而通用爬虫一般只是单单的爬取整个网页，或者是过滤掉 HTML 标签，这样获取的语料带有许多无关信息，对模型训练或文本进一步分析都是额外的噪声，所以文本采集任务更适合使用定制爬虫。

2.3.2 文本切分关键技术

文本切分关键技术是针对中文文本而言的，因为中文文本没有分隔符来区分文本之间的单词，所以需要进行分词处理，将整个文本切割成由单词组成的文本序列，中文分词技术是自然语言处理里非常重要的技术，同时，也是进行文本分析的前提条件，常见的分词算法是基于词典和规则的最大匹配算法，这类算法一般需要有强大的词库用来进行分词，但是，对于词典中不存在的词，这类算法就会失去作用，而且对于存在歧义的文本序列，这类算法同样不能达到满意的效果，于是，出现了基于概率统计的分词算法，其中最经典的是使用 HMM 算法进行分词，HMM 算法可以消除对歧义词语切分不准的问题，而且还可以切分出未登录的词，所以现代的中文分词程序都是结合两种方法进行分词，而且分词效果也十分出色，一般能达到 98% 的准确率。

2.3.3 文本标注关键技术

文本标注一般是在文本分类训练过程中对语料进行标注，形成熟语料，从而可以训练所需的模型，文本标注任务根据需求的不同标注方法也不尽相同，一般的标注方式有三种：人工标注，基于特征词典的文本标注，基于标签的文本标注，以下是对三种标注方式的解释：

(1).人工标注：顾名思义，人工标注就是采用人工的方式对文本进行标注，这种方式的标注成本很高，但是标注效果好，一般是对标注要求很高的需求才会使用人工标注的方式。

(2).基于特征词典的文本标注：基于特征词典的标注一般是采用一个特征词典，根据分词匹配结果计算文本的最终标注结果，特征词典一般包含特征的级性，采用线性叠加的方式计算文本片段，计算结果就是文本的标签值，采用特征词典的文本标注适用于标签比较少的请求，例如情感倾向标注就适合使用该标注方式。

(3).基于标签的文本标注：标签标注是通过文本在源文本中的标签特性进行标注，一般依赖原文本的标签值，例如在爬取评论信息时，根据评论的星级以及点赞次数，可以标注该评论文本的倾向和情感强度，这种标注方式依赖源文本的标签，如果原文本没有

明显的标签就需要结合人工识别的方式指定特定的标签作为标注特征，再结合特征词典的方式会达到很好的效果。

2.4 本章小结

本章介绍了文本分析领域的相关知识点、算法和关键技术，在相关知识点小结中介绍了文本分析中涉及的语言模型、文本表示方法、文本特征抽取等知识点，通过该节的介绍，读者可以对文本分析中出现的知识点有所了解。传统的文本倾向分析算法大多是基于规则和统计学算法提出的，但是随着时代的进步，这些算法已经不能满足现在的业务需求，在第二节中，本文介绍了最新的研究成果，采用神经网络模型作为处理文本倾向分析问题的核心算法，并结合主题模型作为文本特征表示的特征向量，以新的角度处理文本分析相关问题。最后，本章介绍了关于文本分析的关键技术，包括语料采集关键技术、文本切分关键技术、文本标注关键技术。这些技术都是文本分析阶段不可缺少的中间环节。通过本章的介绍，读者可对文本分析领域有一个全面的认识，为之后章节的理解打下基础。

第三章 基于大数据平台的文本处理技术

文本处理任务是一个计算密集型任务，而且通常需要训练的语料库非常庞大，动辄几十 G，上百 G 的语料需要训练，而且随着互联网的更新迭代速度的加快，数据的产生以指数级的规模不断的扩大，单机环境已经无法满足文本数据的处理需求，所以十分需要借助分布式计算能力来提高计算效率，随着大数据技术的发展，大数据计算平台越来越成熟，同时也越来越受到业界的青睐，使用大数据平台来处理海量文本数据已经成为许多企业首选的解决方案，本节主要通过介绍 Spark 大数据平台来阐述使用大数据平台的优势，并给出如何使用 Spark 处理文本数据的步骤。

3.1 Spark 大数据处理平台

Spark 作为现今最流行的内存分布式计算框架，其优点远远不仅仅是作为大数据的计算中心这么简单，它已经形成一套完整的大数据处理生态系统，其中包括 Spark Streaming:用来处理实时数据，Spark SQL:用类 SQL 语句对 Spark 数据进行查询，Spark MLIB:可扩展的机器学习库,包括常用的机器学习算法,Spark Graphx:支持图计算和并行图计算的 Spark 库。

Spark 大数据计算框架是建立在 MapReduce 基础之上的分布式计算框架，但是相较于 Hadoop 简单的 Map 和 Reduce 操作，Spark 提供了一个分布式内存抽象数据集 RDD，针对 RDD 还提供了更加丰富的算子操作，这些操作被划分为转换（Transformations）和动作（Action）。转换操作是对 RDD 进行的用来返回一个新的 RDD 的算子，包括 map、reduce、union、filter、reduceByKey、partitionBy、join、count、sample 等，动作是在数据集上经过一次计算后返回的结果。所有的 Spark 应用都是由驱动程序组成，这些驱动程序用来并行执行用户提交的计算任务，计算任务又由各个单向的转换操作组成，在 Spark 中，所有 RDD 的转换都是惰性求值的，旧的 RDD 数据经过转换后生成新的 RDD 集合，每个 RDD 又可以有多个分区，这样每次 RDD 转换相当于生成一个结点到另一个结点的有向无环图(DAG),由多个 DAG 构成一个 Spark 任务，同时，Spark 对于 DAG 任务进行优化处理，确定阶段、分区、流水线、任务和缓存，数据按照 DAG 的方向流动，并当遇到 Action 类型的算子后进行实际运算，这样数据的流转都在内存中进行，大大提高了计算效率。下图展示了 Spark 的计算模型。

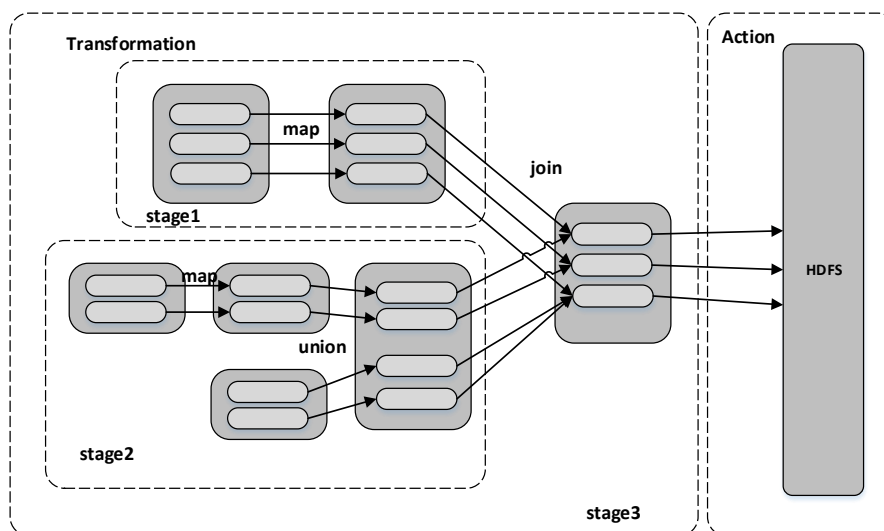


图 3-1 Spark 计算模型图

3.2 Spark 大数据平台的优势

● MapReduce

MapReduce 是一种抽象的编程模型，通过将复杂的数据处理分解为 Mapper 和 Reducer 两个过程，然后将这些分解后的计算 job 分发到几十到几百台服务器上分布式的运行，同时对外隐藏了并发、分布、故障恢复、集群通信等细节问题，从而可以处理大数据集的计算问题。

MapReduce 过程是将复杂问题抽象化，隐藏抽象处理的过程，这些抽象的细节被称为 shuffle 过程，shuffle 过程是 MapReduce 过程中非常重要的一环，当问题被分解为以 Mapper 和 Reducer 构成的有向无环图(DAG)时，各个 job 的最终执行过程是受 shuffle 过程影响的，如果分解合理会大大提高执行的效率，反之，即使经过了 MapReduce 过程可能也不会达到预期的效果。下图中 map 和 reduce 中间的过程就是 shuffle 过程。

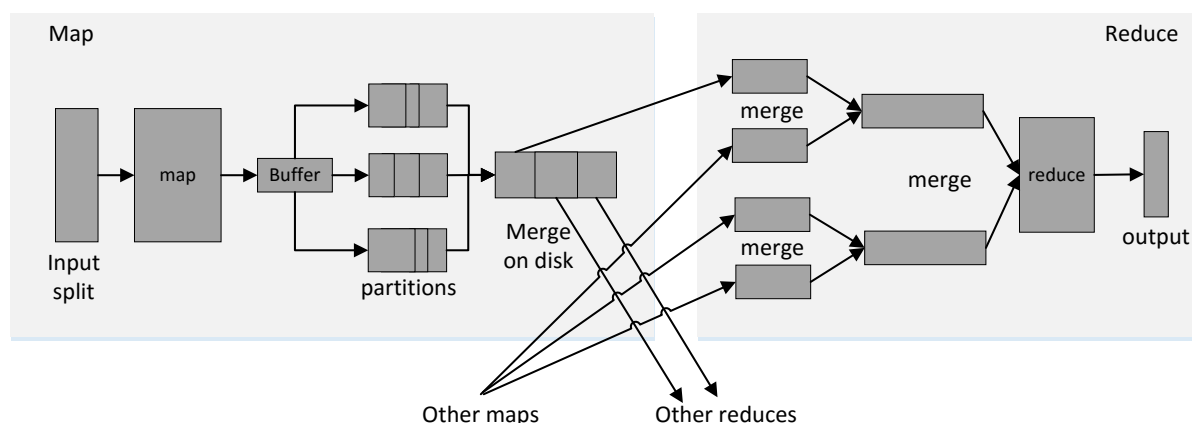


图 3-2 Map-Reduce 计算模型示意图

所以, MapReduce 也存在一定的局限性, 首先 MapReduce 的抽象层次比较低, 只提供 Mapper 和 Reducer 两个操作, 对程序员不够友好, 很难上手写 MapReduce 程序。其次, 一个复杂的任务需要分解为多个 job 进行计算, job 的管理需要开发者自己维护, 如果设计的不够合理会导致效率下降, 而且处理的实际逻辑隐藏在 shuffle 过程中, 不受开发者的管理, 出现问题需要修改 MapReduce 过程, 维护成本很高。最后, Reducer 需要等待所有 Mapper 执行结束后才能执行, 且中间结果需要保存到 HDFS 上, 没有完全利用集群的内存资源。

针对 MapReduce 的局限性, 研究者在 MapReduce 的基础上进行了许多改进, 出现了如 Pig, Cascading, JAQL, OOzie, Tez 等技术, Spark 是其中比较新的一种技术, 而且基于其出色的表现能力, 逐渐成为独立的大数据计算框架。

● 从 MapReduce 到 Spark

Spark 是基于 MapReduce 的大数据内存计算框架, 其主要特点是有一个抽象的分布式内存对象 RDD(Resilient Distributed Dataset), RDD 作为 Spark 编程模型的核心, 具有很多丰富的操作, 这些操作可以分为两类: Transformations 和 Action。Transformations 包括 map, flatMap, filter, union, sample, join, groupByKey, cogroup, ReduceByKey, cross, sortByKey, mapValues 等, Action 包括 collect, count, save, lookupKey 等。

Spark 作为新型的计算模型相对于 Hadoop 来说具有很多优点, 首先, Spark 是基于内存的抽象计算模型, 在性能上比 Hadoop 的 MapReduce 有很大的提升, 2014 年 10 月, Spark 完成了 Daytona Gray 排序测试, 该测试结果显示 Spark 用相对于 Hadoop1/10 的计算资源完成该项测试的耗时比 Hadoop 的 MapReduce 过程的耗时快 3 倍。以下是该次测试的实际结果对比:

	Hadoop MR Record	Spark Record	Spark 1PB
Data Size	102.5TB	100TB	1000TB
Elapsed Time	72mins	23mins	234mins
#Nodes	2100	206	190
#Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150GB/s	618GB/s	570GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	Dedicated data Center, 10Gbps	Virtualized(EC2) 10Gbps network	Virtualized(EC2) 10Gbps network
Sort rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Sort rate/node	0.67 GB/min	20.7 GB/min	22.5 GB/min

图 3-3 Spark 计算框架和 Hadoop 计算框架效率对比

其次, Spark 中的 RDD 都是惰性求值的, 一个 RDD 会转换成新的 RDD, RDD 之间不是强依赖关系, 对于非依赖的 RDD 计算, 并不需要等待其运行结束就可以执行其他运算。所以 Spark 对迭代式计算有更好的支持。最后, Spark 具有丰富的扩展, 其不单单是一个内存计算模型, 还包含交互式 (Spark SQL), 流式 (Spark Streaming), 机器学习 (MLLIB), 图形计算 (GraphX) 等模块, 所以 Spark 已经成为一个大数据计算平台, 许多计算任务都可以在 Spark 上进行运行。下图是 Spark 模块的构成:

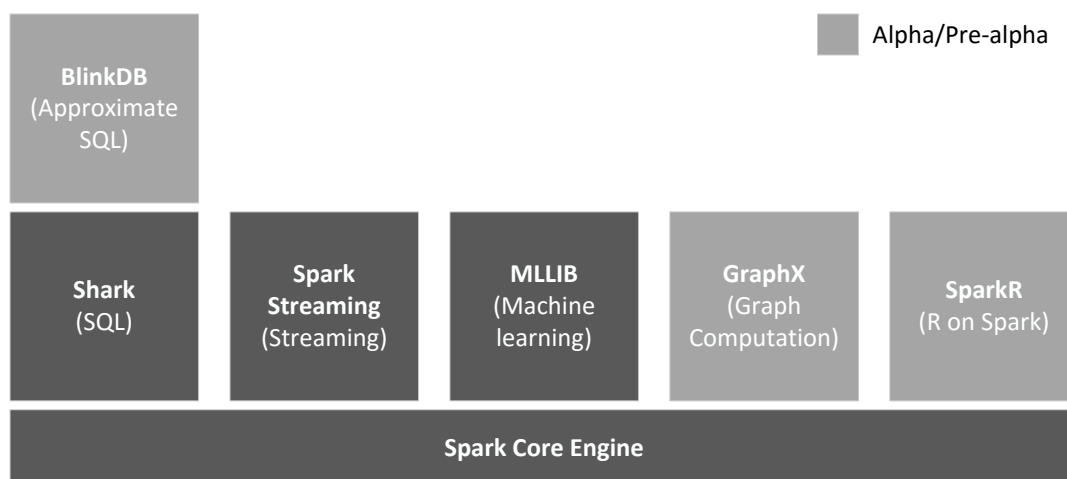


图 3-4 Spark 计算平台的模块构成

3.3 如何使用 Spark 大数据平台处理文本数据

以上两节分别介绍了 Spark 大数据平台和 Spark 的优势, 本节针对如何使用 Spark 大数据平台处理文本数据这一问题进行阐述, 根据第二章介绍的有关文本分析的知识可知, 文本分析主要的计算任务是需要对文本语料进行预处理和模型训练。文本预处理包括文本切分、文本过滤、文本标注等, 文本模型训练包括文本特征训练、文本特征抽取、文本分类模型训练等。通常情况下, 这些文本任务都是通过不同的模块进行分别处理, 而基于 Spark 平台的支持, 这些任务可以结合在一起完成。

首先, 对于初始语料的处理, Spark 提供了针对 Java、Scala、Python 的接口, 可以选择其中的任何一种语言对语料进行处理, 而且 Spark 也提供了对语言内部数据结构和 RDD 相互转换的接口, 所以, 可以通过现有的语言编写好对语料的处理逻辑, 然后根据 Spark 提供的接口, 将处理逻辑转换成 Spark 内部的数据结构, 即 RDD 内存模型, 然后进行分布式的处理。下图是以初始语料集处理为例, 展示如何将文本处理任务运行到 spark 平台上:

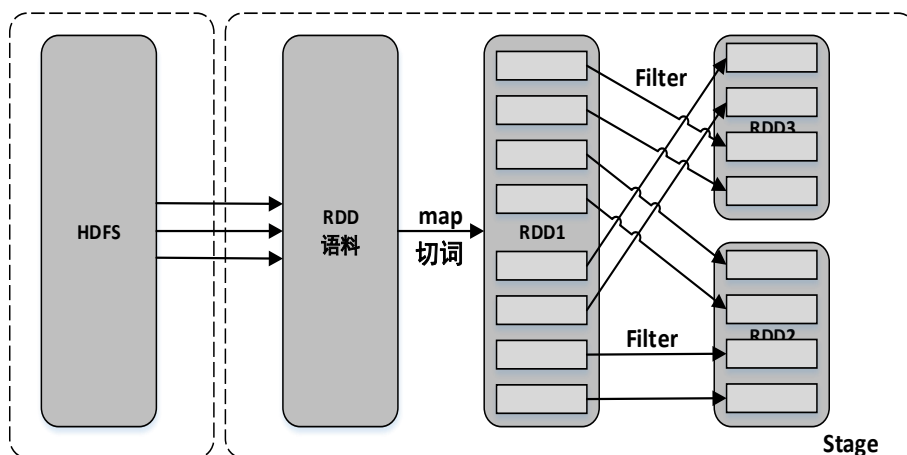


图 3-5 利用 Spark 处理文本数据

初始语料可以保存在 HDFS 上，从 HDFS 中将语料文本导入 Spark 系统，构造初始语料 RDD，通过 map 操作将语料 RDD 切分成词序列保存到新的 RDD 里，这里记作 RDD1，Spark 的 map 操作可以通过函数将 RDD 映射到新的 RDD 上，这里的映射函数就是通过其他语言实现的分词模块，形成的 RDD1 实际上是一个列表，其中每个元素是一个 token 对象，该对象包含三个属性：word, offset, nature。其中 word 指单词本身，offset 为该词在语料中的位置，nature 指该词的词性。根据 nature 我们可以通过 filter 算子获得具有特定特征的词语的集合 RDD2，然后，我们通过停用词词典将切分后的语料过滤，剔除无用的停用词，例如“的，是，了”等，形成待训练的新语料 RDD3，用来后续的文本分析任务。

以上，就是对初始语料的处理，之后我们可以通过利用 Spark 提供的 MLLIB 库对语料进行训练获取语言模型，MLLIB 库提供了丰富的机器学习算法，其中就包括第二章提到的用于进行文本分析的算法，对于没有实现的算法，我们同样可以使用外部的机器学习库来训练语料，然后通过将其转换成 RDD 在进行其他任务。

通过以上的描述，我们知道了使用 Spark 大数据平台处理文本是非常方便的，同时，基于 Spark 出色的表现力，还可以提高文本处理的效率，具有开发方便，性能卓越的优点。

3.4 本章小结

本章介绍了如何应用 Spark 大数据平台处理文本数据，首先，本章介绍了 Spark 大数据平台的相关内容，包括 Spark 的由来、特点、以及计算模型，Spark 作为新兴崛起的大数据内存计算框架，其显露出的强大计算能力和丰富的组件，使得其成为解决海量数据处理问题的首选平台。其次，本章通过和传统的 Hadoop 平台进行比较，来阐述 Spark 平台具有的优势，从计算模型丰富度、速度、组件种类三个方面来看，Spark 比 Hadoop

具有明显的优势，同时也非常适合用来处理文本数据。最后，本章通过一个例子来讲解使用 Spark 处理文本数据的可行性，以及 Spark 作为文本处理平台具有的优势。通过本章的介绍，阐述了基于大数据平台设计中文文本分析系统的大体思路，为读者理解整个系统的设计打下基础。

第四章 基于大数据平台的文本倾向分析系统的设计

本章以 1~3 三章为基础，设计并实现一套基于电影评论的文本倾向分析系统，根据前几章的讲解，文本倾向系统一般涉及文本的获取、文本的切分、文本的标注、文本的训练以及文本的分类，本系统就是围绕文本分析的这 5 方面设计的。同时，在系统设计过程中，采用基于情感词典和标签的融合标注算法来对训练语料进行标注，并设计了基于 doc2vec 和 LDA 的文本倾向分析算法，该算法向较于其他文本倾向分析算法，不仅提高了准确度，而且是无监督的分析算法，同时，本系统借助 Spark 大数据平台，将算法采用并行化的方式运行在集群上，提高算法的并行能力和训练速度。以下是对系统流程的描述：

首先，该系统通过定制化网络爬虫从豆瓣网站上爬取电影的评论信息作为训练语料，评论信息包括评论发表的日期、星级、点赞数、评论文本，并将这些信息以结构化数据的形式保存到数据库中。然后，通过标签（包括星级，点赞数）对评论文本进行正负性分类，对于无法判断正负倾向性的评论，采用情感词典的方式对其进行正负性分类。分类后的文本分为正向评论和负向评论，然后对正负评论文本进行分词处理，去掉停用词，并采用 LDA 算法对文本的主题进行提取，作为整个文本的主题特征。之后，对文本进行 Word2vec 模型训练获取 word2vec 模型，并结合文本的主题特征训练 doc2vec 模型，最后采用 SGD 分类算法训练分类器模型，并得到最终的文本倾向分析模型。最后，通过实验验证该模型的准确度和速度，最终实验表明采用 LDA 和 doc2vec 算法可以提升文本倾向分析的准确度，并通过使用 Spark 并行化算法提升模型训练的速度。

基于以上的系统流程，本章以如下结构组织来阐述整个系统的设计。

4.1 节是基于文本倾向分析系统总体框架设计，主要讲解整个文本倾向系统的总体设计框图和各模块设计细节。

4.2 节是基于 doc2vec 和 LDA 的文本倾向分析算法的设计，该节主要讲解文本倾向系统的倾向分析算法，该算法是系统的核心算法。

4.3 节是中文文本倾向分析系统的并行化实现，主要讲解如何在 spark 上并行化文本倾向分析的各个环节。

4.1 文本倾向分析系统总体框架设计

本系统是针对电影评论倾向分析设计的文本分析系统，其整个设计思路都是与文本分析的各个环节一一对应的，本系统主要流程设计由四部分组成：文本预处理阶段、

文本存储阶段、文本分析阶段和结果展示阶段。处理流程图如下所示：

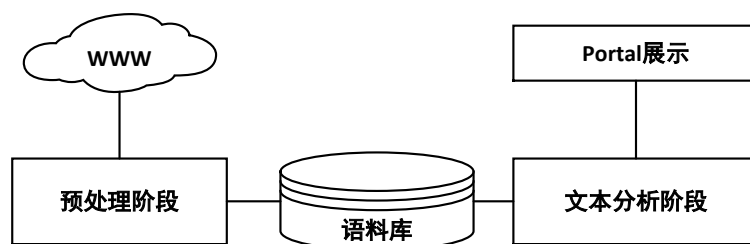


图 4-1 文本分析处理流程示意图

首先系统从豆瓣网站获取电影的评论信息作为初始语料，然后经过对电影信息的预处理后作为训练语料保存到语料库中，之后进入文本分析阶段，其中包括对模型的训练和分类，输出是训练好的文本倾向分类模型，然后通过展示界面展示模型的准确度，并提供接口来演示对文本的倾向性判断。

根据以上所述的系统整体流程，设计本系统的总体框架图如下：

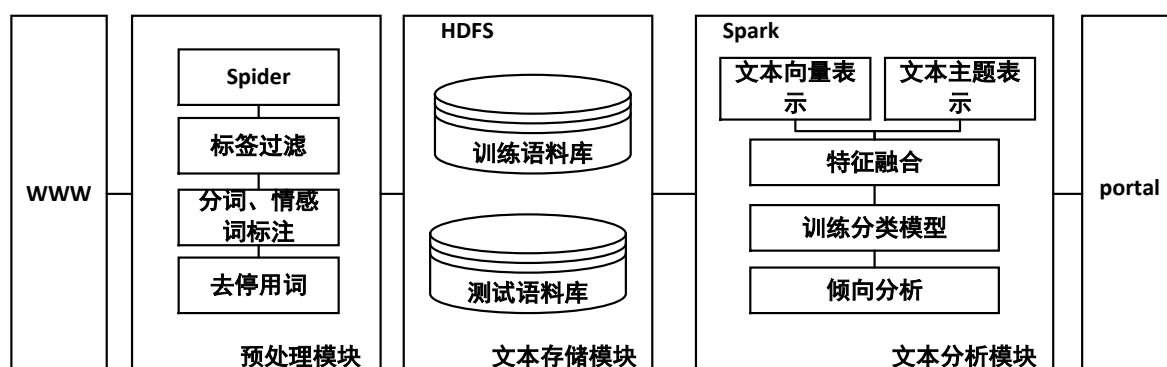


图 4-2 文本倾向分析系统总体框架图

其中系统分为三个主要模块，分别是文本预处理模块、文本存储模块、文本分析模块，以下是对各个模块的功能介绍：

(1).文本预处理模块：文本预处理模块主要负责的是文本的获取和处理，其中文本获取采用定制化 spider 爬虫来获取文本语料，相对于通用爬虫来说，无差别的爬取训练语料，对于文本特征的提取会产生很多噪声，这些噪声会大大降低对文本训练的准确度，从而影响最终的效果，采用定制化爬虫，可以根据网页的特征来爬取需要的语料进行分析，例如对于电影评论信息，需要爬取的内容包括评论主体、点赞数量、评论打分、评论标签等信息，而通用爬虫往往只是简单的将网页标签去掉，剩下文本部分作为训练语料，这样的文本语料粒度非常粗，经过分词和过滤后的效果也往往很难达到要求。而有差别的对这些信息进行分类处理，不仅可以去除语料的噪声影响，而且有利于对文本特征的提取。

文本预处理模块的另一个重要的功能是对爬取语料的切分和标注，对于文本倾向

分析来说，需要对文本中的情感词进行标注，情感词的标注对于之后特征提取权重计算很有帮助，情感词是保存在情感词典中的单词，情感词典往往会包含这些单词的词性、级性、主客观性等属性，通过对文本的情感词进行判断可以获得文本中句子的情感级性，通过对句子的情感级性的判断可以获得该文本段的情感级性，从而对判断整个文本的情感级性有很大的帮助。

(2).文本存储模块：文本存储模块是将文本保存成特殊的格式保存到分布式文件系统中，作为文本分析的输入，为了达到公平性，需要将处理后的文本分为两类，一类是带有情感级性的训练语料，一类是隐去情感级性的测试语料，最终通过对测试语料的分析获得的打分和其实际的打分进行对比来衡量该系统的准确性。

(3).文本分析模块：文本分析模块采用两种文本表示模型来提取文本的特征，一种是基于神经网络语言模型对词进行向量化处理，一种是采用文本主题模型对文本主题进行提取，通过对词向量特征和主题特征进行融合，作为该文本的特征向量，进行分类模型训练，最后得出分类模型进行倾向性分析。

4.1.1 文本预处理模块设计

文本预处理模块包括定制化爬虫模块和分词模块，其模块构成关系图如下所示：

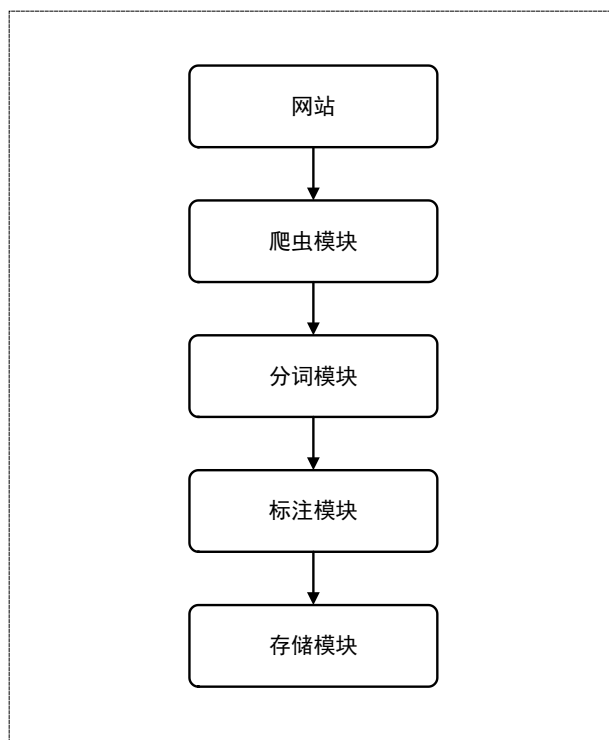


图 4-3 文本预处理流程图

根据系统的要求，爬虫模块的设计需要提供以下功能：1.对特定的页面进行过滤，并提取系统所需的信息，2.爬虫模块还需要能模拟用户行为防止被屏蔽，3.爬虫需

要有自定义的输出接口，可以满足灵活的输出需求。4.爬虫需要保证一定的爬取速度，在不影响原网站正常提供服务的前提下，尽可能快的爬取所需信息。分词模块的设计需要满足的功能包括：1.支持自定义词典的导入。2.支持对切分单词的标注。

由于爬虫模块和分词模块是文本预处理过程中对初始训练语料的准备阶段，对于这两个模块的设计，其目的是为模型训练提供满足要求的熟语料。本系统在开源框架的基础上，分别设计爬虫模块和分词模块，并设计针对电影评论文本倾向分析的标注模块来完成整个文本预处理模块的设计。以下是对各模块的设计：

(1). 基于 scrapy 的定制化爬虫框架设计

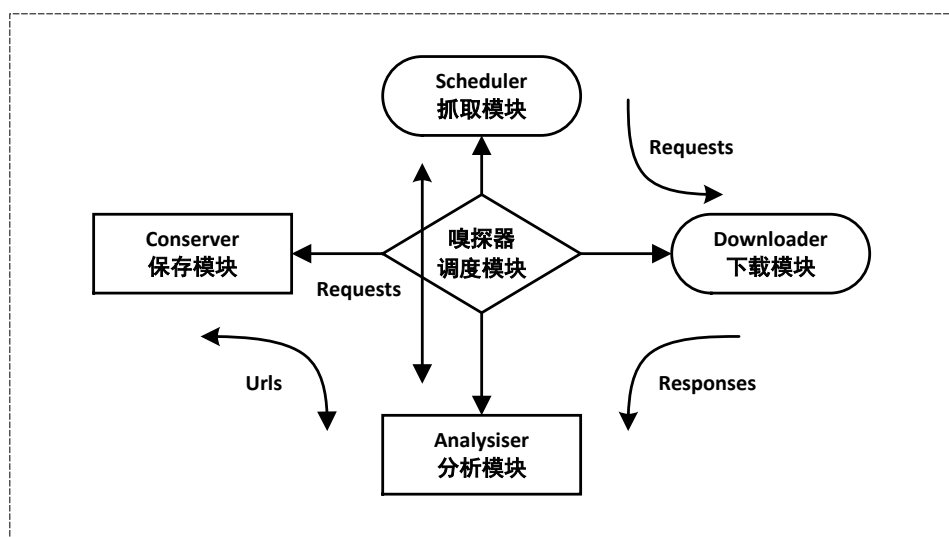


图 4-4 定制化爬虫框架结构图

本系统的爬虫是基于 scrapy 设计的针对豆瓣网站电影评论的定制化爬虫，该爬虫具有的功能包括，1.对特定 HTML 标签内容的获取，2.模拟登录，3.支持模拟用户行为的反屏蔽策略，4.自定义输出数据结构。爬虫的总体结构如下图所示：

由上图可知，爬虫主要由 5 个模块组成，其分别是：调度模块、抓取模块、下载模块、分析模块、存储模块。其中各模块的作用如下：

- 调度模块：负责调度其他模块，其他模块将数据发送给调度器，调度器根据请求类型将数据发送给下一个模块
- 抓取模块：抓取模块主要负责发起 http 的 request 请求，并将 response 结果发送给下载模块。
- 下载模块：下载模块主要负责处理接收的 response 结果，并将结果发送给分析模块。
- 分析模块：分析模块获取到下载模块发送来的内容，分析网页内容，并将满足要求的内容发送给存储模块，然后提取需要抓取的其他链接，并发送给抓取模块，完成下一轮的抓取任务。

- 存储模块：将爬取到的内容以固定的格式保存到数据库或文件中。

根据爬虫的功能结构图，可以得到如下爬取豆瓣电影评论的流程图：

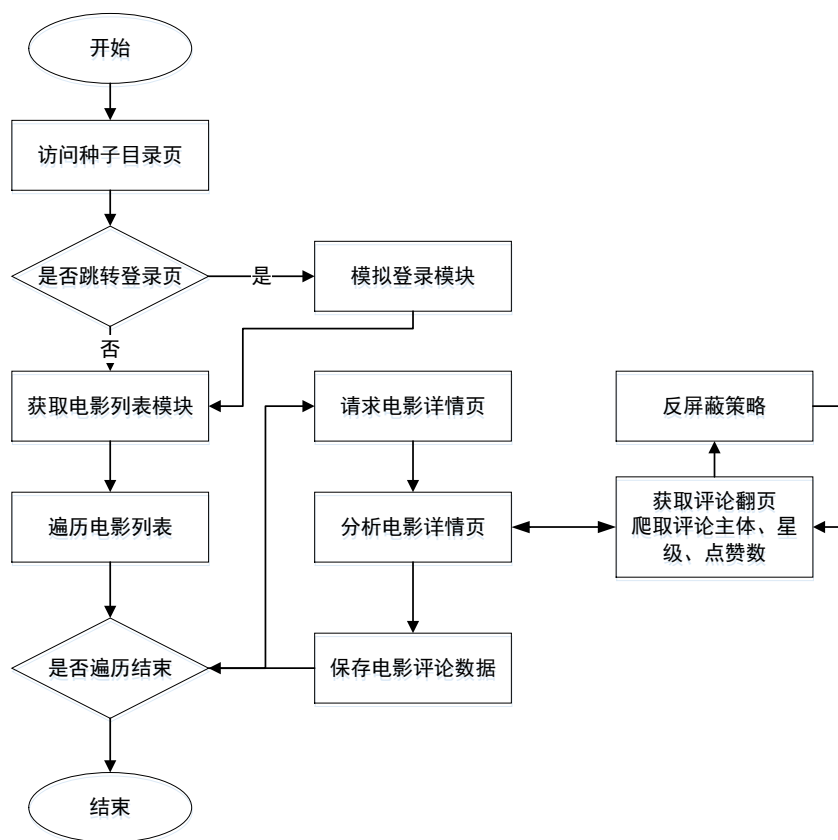


图 4-5 爬取豆瓣电影评论流程图

由于本系统是针对豆瓣网站电影评论的爬取，所以需要首先完成登陆功能，在爬虫初始化阶段，初始化 http 请求头中的 cookie 字段为事先保存好的登陆信息，然后通过抓取模块向登陆 url 发送请求，在分析模块中获取到服务器端 http 头部返回的 cookie 信息，并保存起来，这样就完成了模拟登陆功能。

然后根据电影种子 url 列表，爬取需要分析的电影评论，在爬取过程中，需要在 http 头里携带上刚才保存的 cookie 信息，保持登陆状态。由于豆瓣网站设置了防范爬虫的策略，所以在爬取过程中会导致爬虫被重定向到验证码输入界面，通常情况的解决方案是，采用验证码识别工具来破解验证码，这种方式的弊端是准确率低且影响爬虫效率。

在本系统中，采用模拟用户行为策略来避免爬虫遇到类似情况，模拟用户行为策略主要思路是：这种情况的发生主要是由于爬虫过度频繁的请求某个域名，导致超过服务器端的警报阈值，然后，服务认定该行为恶意访问，所以会把请求重定向到验证码输入界面。一种解决方案是，设置爬虫的爬取间隔，搁固定时间进行爬取，例如用户一般访问下个页面的频率是 3s 一次，则设置爬虫的爬取间隔为 3s。采用这种方式可

以降低爬虫被屏蔽的概率，但是仍然会导致爬虫不可用，且这种方式使得爬虫的效率大大降低。本系统采用的解决方案是：模拟用户的访问行为，根据用户访问频度，设置一个访问间隔区间，例如是 1s-10s，然后采用随机取样的方式从区间中取得一个访问间隔，并用该间隔作为爬虫的爬取某一链接的间隔，采用这种方式可以很好的模拟用户随机访问页面的行为，很具有迷惑性，而且爬取的间隔不是固定值，爬取效率受随机采样的期望决定，可以根据不同的需求动态的改变采样方式，从而动态的调节爬取速度。

爬虫获取到 html 页面后会调用分析模块对页面中的评论信息进行抽取，这里需要利用配置好的 Xpath 策略来获取有用的信息，例如评论文本、评论星级、评论点赞数、评论日期，然后将这些数据封装为一个 item 发送给存储模块，存储模块会将 item 保存到数据库或文件中，这样就可以得到电影评论语料。

(2). 基于 jieba 分词的分词模块设计

分词模块是用来处理爬虫爬取的评论语料，对于中文文本来说，需要使用分词模块来对文本进行切分，本系统采用 jieba 分词作为分词库，jieba 分词是开源的分词库，其支持自定义词典，对于网络评论来说，其中包含大量的网络用语，如果用普通的切词词典进行切分，会导致许多网络用语切分不正确，所以，需要扩充其自带的词典进行切词，同时，本系统是解决文本倾向性的，对于情感词比较敏感，所以还需要使用情感词典扩充词库，而且情感词库还用于对文本的正负性标注。所以，基于对本系统的需求，切词模块的流程图如下所示：

首先导入网络词库和情感词库，然后对文本进行切分生成切分语料，最后去掉停用词，生成过滤后的切分语料。

(3). 基于情感标注和标签融合的文本标注方法的设计

切分后的语料需要进行正负性标准，从而生成正负训练集。本系统采用自动化标注方法，对于豆瓣电影评论，评论属性中包含 1-5 级的打分，分数越低代表用户对该电影越不喜欢，相反，分数越高，代表用户越喜欢该电影。基于对常识的认识，我们认为分数的高低也部分反映了用户对电影评论的正负倾向，分数越高的用户越容易给出正面评论，分数越低的用户越容易给出负面评论。由于有分数标签，我们可以按照分数将文本分为 1-5 类，情感的强烈程度依次递增。本文采用如下算法对评论文本进行正负性标注：

定义 D_i 为第 i 个评论的类别， N_i 为第 i 个评论中负向情感词个数， P_i 为第 i 个评论中正向情感词的个数，则对于第 i 个评论文本：

- 如果 $D_i = 5$ 或 4，则该评论倾向性为正
- 如果 $D_i = 1$ 或 2，则该评论倾向性为负
- 如果 $D_i = 3$ 且 $N_i > P_i$ 则该评论倾向性为负
- 如果 $D_i = 3$ 且 $N_i < P_i$ 则该评论倾向性为正。

采用如上的算法，可以将训练语料分为正向语料和负向语料，正负向语料经过特征提取，训练后就可以得到模型。

4.1.2 文本存储模块设计

文本存储模块主要作用是将爬取的评论语料和预处理后的文本存储起来，为进行文本分析做准备，本系统采用 HDFS 作为文本存储的文件系统[46]，将正负文本保存到 HDFS 上，并在数据库中保存文件的元信息，当用于生成模型时，文本分析模块从数据库中读取文本元信息，然后从 HDFS 上下载需要训练的文本到本地进行训练。

HDFS 一般作为 Spark 大数据平台的底层存储系统[47]，Spark 有专门的接口从 HDFS 上读取文本并将其转换成 RDD，同时也可以将 RDD 持久化到 HDFS 上作为中间结果保存。HDFS 是主从式架构，一个 HDFS 集群由一个单一的 NameNode 和一定数目的 Datanodes 组成。其中，NameNode 作为 Master 节点，其主要负责维护文件系统的命名空间，同时，提供客户端访问文件的调度策略。Datanodes 作为存储节点，通常每一个 Datanode 对应集群中的一台物理节点，用来管理保存在其上的文件数据。HDFS 文件系统对外提供一个命名空间并允许用户将数据存储到文件中。文件以数据块的方式被存储在一组数据节点上，下图是 HDFS 的架构图：

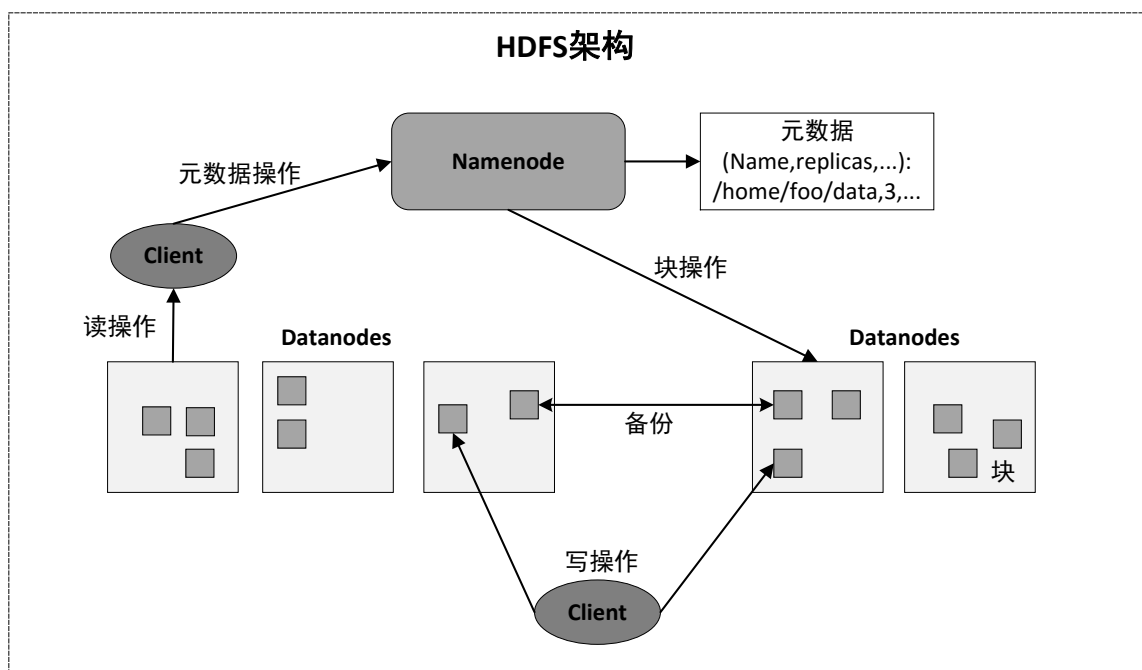


图 4-6 HDFS 架构图

文本存储模块是基于 HDFS 设计的存储模块，采用小文件存储的方式[48]，爬虫将爬取到的评论文本以电影类别分类并保存到 HDFS 文件系统上，然后将评论文件在 HDFS 上的路径元信息保存到数据库中，并在数据库中保存该文件的其他信息例如电影名、类别、爬取时间、评论条数等信息，同时，文本预处理模块会通过数据库获取已

经爬取的评论信息，并构造用于文本倾向分析的语料库，构造完成的语料库会以固定目录结构保存到文本存储系统上，文本存储模块图如下图所示：

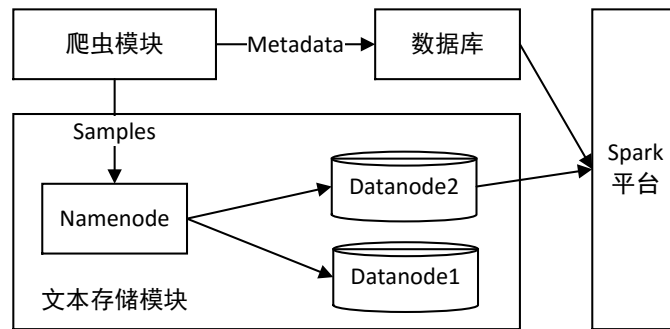


图 4-7 文本存储模块结构图

以 mysql 作为评论文本的数据库信息，其中保存评论信息的数据库表如下所示：

表 4-1 评论信息数据库表

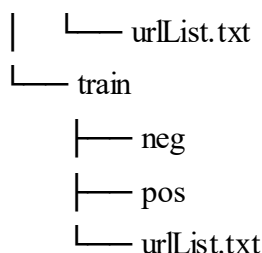
列名	类型	是否主键	是否为空	说明
ID	Int	是	否	主键
MovieName	Varchar	否	否	电影名字
Type	Int	否	否	电影类型
CrawlDate	DateTime	否	否	爬取时间
AveScore	Int	否	否	星级
CommentsCount	Int	否	否	评论数
CommentsFilePath	Varchar	否	否	评论文本在 HDFS 上的路径
Tag	Int	否	否	是否已经被处理

在爬虫爬取过程，文本预处理模块会同时处理已经保存好的评论文本，并将处理后的评论文本保存到语料库中，语料库目录机构是按照一定规律组织的，其目的是为训练模块提供可用的训练语料并用于持久化训练好的模型数据，其保存在 HDFS 上，语料库的目录结构如下：

```

├── douban.vocab
├── models
├── README
├── test
│   ├── neg
│   └── pos

```

其中, `douban.vocab` 是出现在评论里的单词构成的词典, `models` 用于保存训练好的模型, 供之后使用, `test` 和 `train` 分别保存用于训练和用于测试的语料, 其中 `neg` 和 `pos` 文件夹内保存经过标注后的负向评论文本和正向评论文本, `urlList.txt` 文件内保存评论文本所对应的 `url`。

4.1.3 文本分析模块设计

文本分析模块是用来对文本建模的模块, 本系统采用的文本建模方式是采用文本向量表示和文本主题表示融合的方式来训练文本模型, 即吸取了基于统计的语言模型优点有吸取了基于神经网络的语言模型优点, 文本分析模块的模块流程图如下所示:

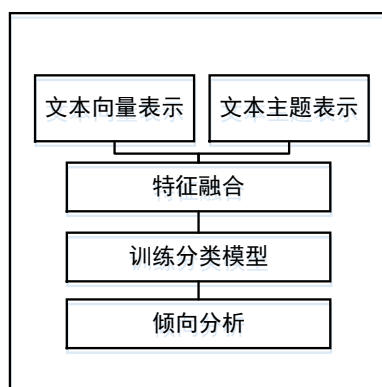


图 4-8 文本分析模块结构图

其中, 文本向量表示采用 `doc2vec` 算法, 文本主题表示采用 `LDA` 算法, `doc2vec` 算法是对文本片段的向量化, 其可以用来表示文本的特征, 文本主题表示采用 `LDA` 算法, `LDA` 算法是一种文本主题模型算法, 通过提取出文本的主题对文本进行稀疏表示, 其也可以作为文本的特征模型, 通过对这两个向量进行融合可以得到文本的特征向量, 然后在通过使用 `SGD` 分类模型分类, 最后得到文本的分类模型, 该模型就是最终用于倾向分析的语言模型。

4.2 基于 `doc2vec` 和 `LDA` 的评论文本倾向分析算法设计

传统的文本倾向分析算法是通过使用级性词典来确定文本倾向性, 通过判断情感词、

否定词、程度副词来计算整个语句的情感偏向，例如“高兴”和“很高兴”中通过程度副词“很”的修饰可以判断情感强度的提高，同时“很高兴”和“很不高兴”中，通过“不”否定词的出现可以判断情感发生逆转。这种基于情感词典判断文本倾向的方式很大程度上是依赖于情感词、否定词和程度副词的词法搭配规则，而搭配规则是无法自动识别的，需要人工配置，同时，相同的词语不同的搭配有时也会出现不同的情感程度，例如“不很高兴”和“很不高兴”出现的情感词、否定词、程度副词相同，但是情感表达确不相同。

随着机器学习的发展，采用文本分类的思路解决文本倾向问题受到人们的关注，一般的思路是对文本进行数字化表示，然后通过分类算法对文本进行分类，常用的分类算法包括逻辑回归，K-means 等，同时，对文本的数字化表示决定了分类效果的好坏，常用的文本表示算法有 bag-of-words 和 bag-of-n-grams，这种方法解决了一部分文本分类问题，但是，对于文本倾向分析，不仅仅是简单的二分类过程，还需要考虑文本之间的词语搭配关系，词与词，词与句之间的关系，采用如上的文本分类思路不能够达到很好的准确率。

本文采用 word2vec 和 lda 作为文本表示算法，充分的考虑了文本之间的搭配特征，并用 doc2vec 算法训练文本获取文本的向量表示，然后通过 SGD 分类算法对文本进行分类，通过实验结果显示，使用该融合算法可以使得文本倾向分析的准确率提高 20%。

4.2.1 基于 LDA 的文本主题特征提取

基于统计的向量空间模型是文档 x 单词矩阵模型，即文档作为行，词表作为列构成的矩阵来表示整个语料库，其中单词的表示可以通过 TF-IDF 值表示。如下图所示：

$$\begin{array}{c} \text{doc}_1 \\ \text{doc}_2 \\ \vdots \\ \text{doc}_n \end{array} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

其中， a_{ij} 是表示单词在文档中的特征值，一般用 TF-IDF 作为该特征值，采用这种方式表示语料库存在很多缺点，这种表示方式会使得该矩阵十分的稀疏，因为对于评论文本来说，doc 中包含的单词一般不会很多（大概是 0-500 个单词），而矩阵的列宽是词表宽度，词表的范围一般几千维左右，这样的表示方式导致列的维度很大而且其中很多项为 0。其次，这种表示方式导致单词与单词之间的信息丢失，只是考虑了单词与文档之间的关系，训练得到的特征模型不具有代表性。

另一种基于统计的向量空间模型是文档 x 主题矩阵模型，该模型的思路是，将文档 x 单词矩阵模型分解为文档 x 主题模型和主题 x 单词模型的矩阵积，如下图所示：

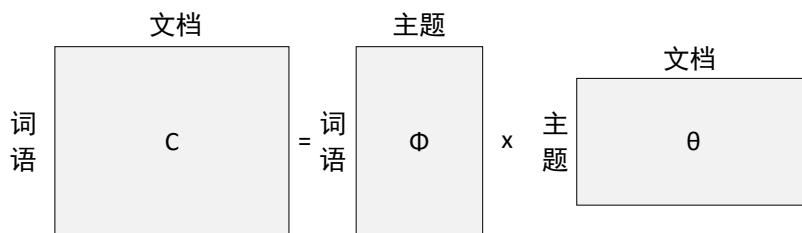


图 4-9 主题模块示意图

其中，一篇文档的生成，即文档词语分布的获得是由文档主题分布和主题词语分布所决定的。通过第二章的阐述，可以知道 LDA 是建立在该主题模型上的。

在本系统中我们通过 LDA 算法来提取文档主题特征，即训练获得文档 x 主题分布。由 LDA 的联合概率密度函数：

$$p(\theta, z, w | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta)$$

可知，隐含参数 θ 是文本的主题概率分布的 Dirichlet 共轭分布，我们可以通过训练 LDA 模型获得参数 θ ，参数 θ 是一个矩阵向量其对应文本的文档 x 主题矩阵分布，即对于一篇文档，我们可以得到该文档主题的分布情况，我们使用文档 x 主题概率分布作为语料的特征表示，即对于 N 篇文档，我们得到 N 个向量 $\vec{\theta}_i$ ，作为标识文档的特征向量，在本系统中，文档对应的是评论文本，即对于每一条评论文本，我们都得到与之对应的一个特征向量，该特征向量标识了该评论文本的主题分布。

4.2.2 结合主题特征的 doc2vec 模型训练

上一节中我们获得了评论文本的特征向量，该节我们通过 doc2vec 算法[16]并结合主题特征来获取文本表示模型，doc2vec 算法是在 word2vec 算法[32]的基础上对文本的向量表示算法，其可以将一段文本片段或文档向量化，根据第二章提到的 word2vec 算法，其训练词向量的主要原理如下图所示：

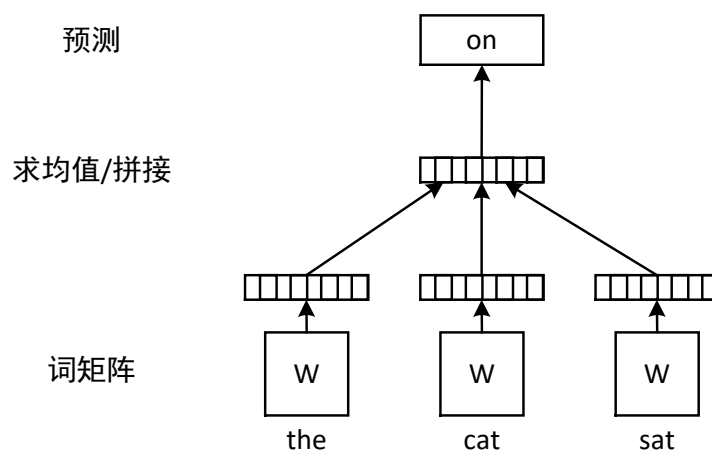


图 4-10 DBOW 模型示意图

其任务是通过给定的词矩阵预测下一个词的概率，在算法中，每个词被映射成一个唯一的向量，分别代表词向量矩阵 W 中的一列。该列的位置代表了该词在语法结构中的位置，这些列向量的加和会作为预测下一个单词的特征向量。将这一思路引申到段，就可以得到段向量(Paragraph Vector)，即 doc2vec 算法。该算法的主要思想是：在 word2vec 的基础上引入段向量，即将每段文本映射成一个唯一的向量，代表段矩阵 D 中的一列，同时，同样将词语映射成列向量组成词矩阵 W 。同理，通过对段向量和词向量的求和取均值来作为预测下一个单词的特征向量，doc2vec 的原理如下图所示：

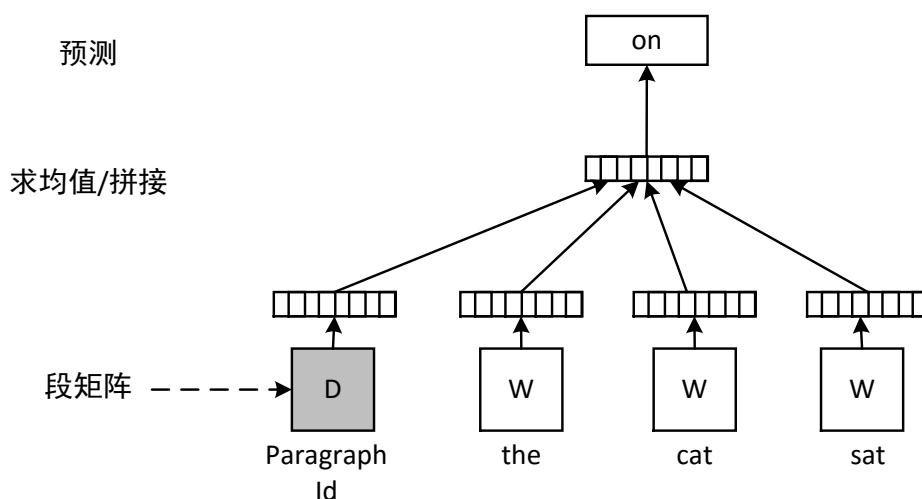


图 4-11 PV-DW 模型示意图

由上图可得，doc2vec 算法与 word2vec 算法的区别是增加了一个段矩阵 D 来训练样本，增加的段向量可以认为是一个新的单词，其可以类比为是当前文本片段的标记，或者是当前段落的主题，该模型为 PV-DM(Distributed Memory Model of Paragraph Vectors)，与 word2vec 算法类似，doc2vec 算法还构造了一个与 PV-DM 模型相对应的模

型 PV-DBOW(Distributed Bag of Words version of Paragraph Vector), 该模型是通过算法从段落中随机抽样并预测单词(如下图所示)。通过这两个模型可以得到段向量: 段向量由两个向量组成, 这两个向量分别是由 PV-DM 训练得到的 paragraph vector 和由 PV-DBOW 训练得到的 paragraph vector。

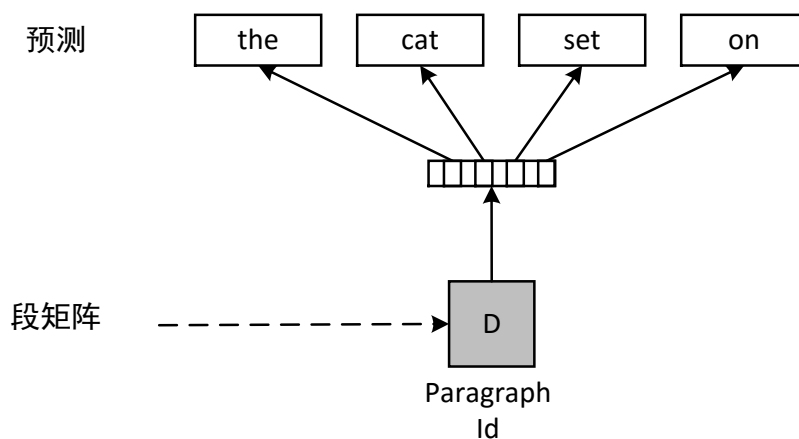


图 4-12 PV-DBOW 模型示意图

通过训练 paragraph vector 可以得到评论文本的向量表示, 同时, 由上一节的 LDA 算法可以训练出文本的主题向量, 对于评论文本来说 paragraph vector 衡量了文本中词与词之间关系的综合, 如果两句话的在语义上相近, 例如“狗趴在地上”和“猫趴在地上”, 那么这两句话所得到的 paragraph vector 也相近, 同时, lda 衡量了文本中的主题分布, 如果两个文本的主题比较相近, 则其主题分布也很相近, 由于这两个向量都表示文档的特征, doc2vec 更倾向于文本的情感, lda 跟倾向与文本的主题

通过以上思路可以得到结合主题特征的 doc2vec 训练算法如下：

输入：

Corpus //表示待处理的语料集合
Models.d2v //训练好的 Doc2vec 模型
Models.lda //训练好的 LDA 模型
Dictionary //训练语料库词典

输出：

```
SentVecs    //表示融合后的文本特征向量，既 $\overrightarrow{d_i^{M+N}}$ 
1  for topic_id in models.lda:    //遍历主题模型中的主题
    //根据词向量和词-主题分布计算每个主题的主题向量
2      for word_id, prob in models.lda[topic_id]:
3          topicVec += prob*models.d2v[Dictionary[word_id]]
4      topicVecs.append(topicVec)
5  for sentence in Corpus:    //遍历语料的句子(既评论文本)
    //根据主题向量和主题-句子分布计算句子向量
6      for topic_id, topic_prob in models.lda[sentence]:
7          sentVec += topic_prob * topicVecs[topic_id]
    //将句子的 d2v 表示和 lda 表示组成新的向量
8      sentVec = [Models.d2v[sentence], sentVec]
9      SentVecs.append(sentVec)
10 return SentVecs    //求得所有语料的融合特征表示向量
```

其中 SentVecs 是融合后的文本特征向量，其表达公式如下：

$$\overrightarrow{d_i^{M+N}} = (\overrightarrow{p_i^M}, \overrightarrow{t_i^N}) \quad i \in C$$

其中，C 是语料库中评论个数，向量式如下： $\overrightarrow{d_i^{M+N}}$ 表示第 i 个评论的向量表示，其维度为 M+N，向量 $\overrightarrow{p_i^M}$ 表示第 i 个评论的 doc2vec 向量，其维度为 M。向量 $\overrightarrow{t_i^N}$ 表示第 i 个评论的 lda 主题向量，其维度为 N。用 $\overrightarrow{d_i^{M+N}}$ 向量作为评论文本的特征向量，考虑了文本的语义相似的和主题相似度，将其作为文本分类特征，是完成文本倾向分析任务的前提

4.2.3 基于 SGD 的文本倾向分类

获得了文本的特征向量后，需要对文本进行分类验证，本系统采用随机梯度下降法（Stochastic Gradient Descent SGD）作为文本分类器，SGD 是一个简单有效的方法，用于判断使用凸 loss 函数（convex loss function）的分类器（SVM 或 logistic 回归）。即使 SGD 在机器学习社区已经存在了很久，到它被广泛接受也是最近几年。

SGD 被成功地应用在大规模稀疏机器学习问题上（large-scale and sparse machine learning），经常用在文本分类及自然语言处理上。假如数据是稀疏的，该模块的分类器可以轻松地解决这样的问题：超过 10^5 的训练样本、超过 10^5 的 features。

SGD 的优点是：

- 高效
- 容易实现

SGD 的缺点是：

- SGD 需要许多超参数：比如正则项参数、迭代数。
- SGD 对于特征归一化（feature scaling）是敏感的。

对于文本分类任务适合使用 SGD 作为其分类器，原因是文本训练出的特征一般都是高维向量，同时每个维度都是经过归一化的概率值，所以本系统采用 SGD 作为电影评论文本分类的分类器。

4.3 文本倾向分析算法的并行化实现

对于海量的训练语料，采用单机的训练方式无法满足训练速度，这时就需要借助大数据平台将训练算法进行并行化处理[49][50]，本节是针对豆瓣电影评论倾向分析系统中相关算法的并行化实现，主要从三个方面来提高整个系统训练模型的效率，首先，在文本预处理方面对于文本的切分处理可以将其移植到 Spark 平台上进行，将切词处理并行化。其次，在训练 LDA 模型时，可以借助 Spark 平台的 MMLIB 库中 GraphX 接口通过图的方式对训练模型进行并行化处理。再次，对于 doc2vec 算法，其训练过程可以通过 MapReduce 操作进行重构，然后通过 Spark 平台完成其并行化实现。最后，通过实验结果分析来说明并行化处理相对于非并行化时训练速度有很大的提高。

4.3.1 针对文本分词处理的并行化实现

文本分词其实质是一个 Map 过程，是 Map 过程就可以通过并行化来提高文本的切分速度，本文采用 Spark 作为并行化框架，将文本切分通过 Map 操作分布到集群上进行，主要的实现方式如下图所示：

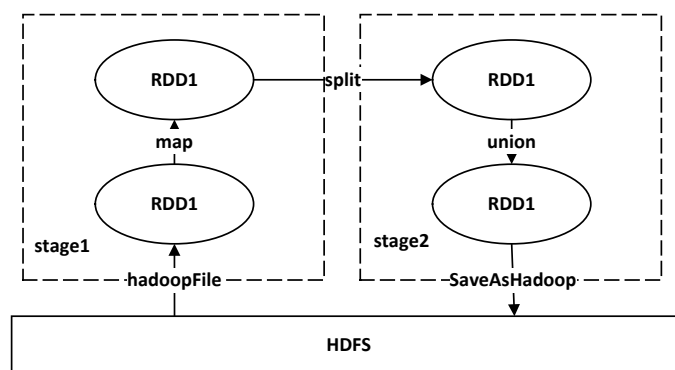


图 4-13 中文分词并行化实现 Spark 程序流转图

文本分词的并行化过程主要分为以下几步：

- (1).将文本从 HDFS 导入到 Spark。
- (2).以句子为基本单位将文本转换为 RDDList
- (3).将 RDDList 使用 map 方式映射为切分后的 RDDList
- (4).将切分后的 RDDList 通过 join 合并为一个 RDD

整个操作的执行过程在 Spark 上是以 Driver 的形式运行的，Spark Driver 的运行过程如下图所示：

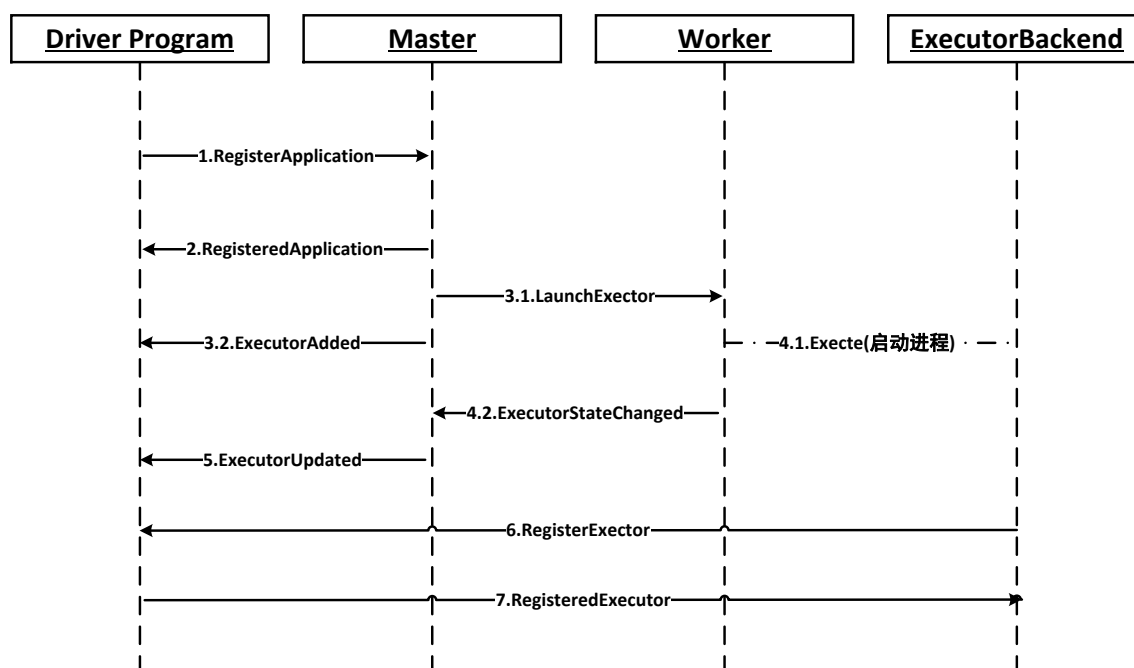


图 4-14 Spark Driver 运行时序图

- (1) Driver 通过 AppClient 向 Master 发送了 RegisterApplication 消息来注册 Application。
- (2) Master 收到消息之后会发送 RegisteredApplication 通知 Driver 注册成功，Driver 的接收类还是 AppClient。
- (3) Master 接受到 RegisterApplication 之后会触发调度过程，在资源足够的情况下会向

Worker 和 Driver 分别发送 LaunchExecutor、ExecutorAdded 消息。

(4) Worker 接收到 LaunchExecutor 消息之后, 会执行消息中携带的命令, 执行 CoarseGrainedExecutorBackend 类(图中仅以它继承的接口 ExecutorBackend 代替), 执行完毕之后会发送 ExecutorStateChanged 消息给 Master。

(5) Master 接收 ExecutorStateChanged 之后, 立即发送 ExecutorUpdated 消息通知 Driver。Driver 中的 AppClient 接收到 Master 发过来的 ExecutorAdded 和 ExecutorUpdated 后进行相应的处理。

(6) 启动之后的 CoarseGrainedExecutorBackend 会向 Driver 发送 RegisterExecutor 消息 Driver 中的 SparkDeploySchedulerBackend (具体代码在 CoarseGrainedSchedulerBackend 里面) 接收到 RegisterExecutor 消息, 回复注册成功的消息 RegisteredExecutor 给 ExecutorBackend, 并且立马准备给它发送任务

最后, CoarseGrainedExecutorBackend 接收到 RegisteredExecutor 消息之后, 实例化一个 Executor 等待任务的到来。

通过以上处理后, 分词任务被分解为 MappedRDD、FlatMappedRDD、MappedRDD 和 ShuffledRDD 四个 RDD 的转换过程。然后通过将分词 job 提交到 Spark 上来触发 Spark 对 RDD 的分布式计算。

4.3.2 针对 LDA 算法的并行化实现

LDA 实现的最终目的是估计文档-主题分布和主题-词分布参数, 从而获得文档的[文档-主题]矩阵和[主题-词]矩阵, 在并行化过程中, 我们将文本和词映射为图关系, 其中文档和词作为图的结点, 词与文档关系作为边, 词频作为边的权值。通过这种方式将语料库转化成为带权图, 同时, Spark 中的 RDD 操作也是处理有向无环图, 可以将 LDA 算法通过图遍历的方式并行化。

LDA 的参数是通过 gibbs 采样估计模型参数的, 我们通过图的方式存储矩阵 n_m^k, n_t^k , 其中, n_m^k 记录第 m 篇文档中第 k 个主题拥有词的数目, n_t^k 记录第 k 个主题拥有词 t 数目, 这两个矩阵以图的邻接矩阵形式存储, 其中矩阵中各个元素的值表示词频。当通过 gibbs 采样完成参数估计时, 其实质是对图的遍历, 通过多次遍历图来完成多次迭代求得 LDA 的主题分布和词分布, 在计算过程中, LDA 模型矩阵、语料矩阵都被视作图的邻接矩阵, 这样就可以将 LDA 算法高效的映射到图上, 完成算法的并行化。

本系统主要使用 Spark 的 Graphx 作为 LDA 算法的图结构，我们采用图来存储主题-词模型，并可以将其分布到各个节点上，其训练 LDA 的过程如下所示：

输入：

- w_m 第 m 篇文档的词向量(未作词频统计)
- α 超参数
- β 超参数
- k 主题个数

全局数据：

- n_m^k 记录第 m 篇文档中第 k 个主题拥有词的数目
- n_k^t 记录第 k 个主题拥有词 t 数目
- n_m 记录第 m 篇文档拥有词的数目
- n_k 记录第 k 个主题拥有词的数目

输出：

参数 θ 估计 文档-主题分布矩阵

参数 α 估计 主题-词分布矩阵

- 1 对 n_m^k 、 n_k^t 、 n_m 、 n_k 置零 //初始化计数器
- 2 for all documents $w_m(m \in [1, M])$ do
- 3 for all words n in w_m do
- 4 $z_{m,n} = k \sim Mult(1/k)$ //为 w_m 中第 n 个词选择主题号，Mult 是多项式分布
- 5 $n_m^k += 1$ // $k=z_{m,n}$
- 6 $n_m += 1$
- 7 $n_k^t += 1$ // t 为第 n 个词在文档词频矩阵中索引号
- 8 $n_k += 1$
- 9 while 参数未收敛并且没有达到最大迭代次数 do //执行迭代过程
- 10 for all documents $w_m(m \in [1, M])$ do
- 11 for all words n in w_m do
- 12 $n_m^k -= 1$; $n_m -= 1$; $n_k^t -= 1$; $n_k -= 1$ //消除词 n 当前选择主题 k 的影响
- 13 $k = p(z_i | z_{-i}, w)$ //消除词 n 影响下，为词 n 选出主题 k ,依据 Gibbs 采样公式
- 14 $n_m^k += 1$; $n_m += 1$; $n_k^t += 1$; $n_k += 1$ //更新统计信息

4.3.3 针对 Doc2vec 算法的并行化实现

大多数基于 Spark MMLIB 的机器学习模型都是并行化训练过程，将训练参数保存到 driver 程序中，并广播给 worker 节点。这类工作往往是基于模型自身相对较小，可以将模型保存到单个节点的内存中，而训练集往往很大无法在单个节点保存，需要保存成 RDD 进行分布式操作。但是，对于 Doc2vec 模型而言，其不符合这种场景，因为 Doc2vec 算法的模型参数规模与训练集中单词个数是成正比关系的。

同时，有学者提出了并行化词向量训练过程[51]，但是主要是并行化采样方式。在本节中，基于 Doc2vec 模型的特殊性，并受到[52][53]的启发，提出一种针对 Doc2vec 模型算法计算过程的并行化方法，Doc2vec 算法是通过训练样本来学习对文档（句子、段落）的向量表示，一个包含 100 万文档的训练集，训练维度为 300 的向量，需要计算的向量空间是 30000 万（ $300 \times 1000,000$ 的矩阵）。但是，在训练过程中，向量的更新是根据其所处行独立更新的，即每个文档的向量在训练过程中独立更新，不会涉及其他行。因此，可以通过压缩模型需要的参数来实现并行化计算，即每个计算部分（通常是句子、段落）只保留与其相关的参数。对于 Doc2vec 算法，其使用梯度下降算法来更新参数，我们可以将梯度下降的过程分解为 gradient 和 descent 两个过程，将数据的 gradient 计算过程分布到其相关的计算部分（可以是 RDD Partition），然后通过 Master Model 进行 descent 计算来更新训练参数，这样，通过各计算节点分布式的执行 gradient 过程来提高算法的训练速度。整个并行化过程如下图所示：

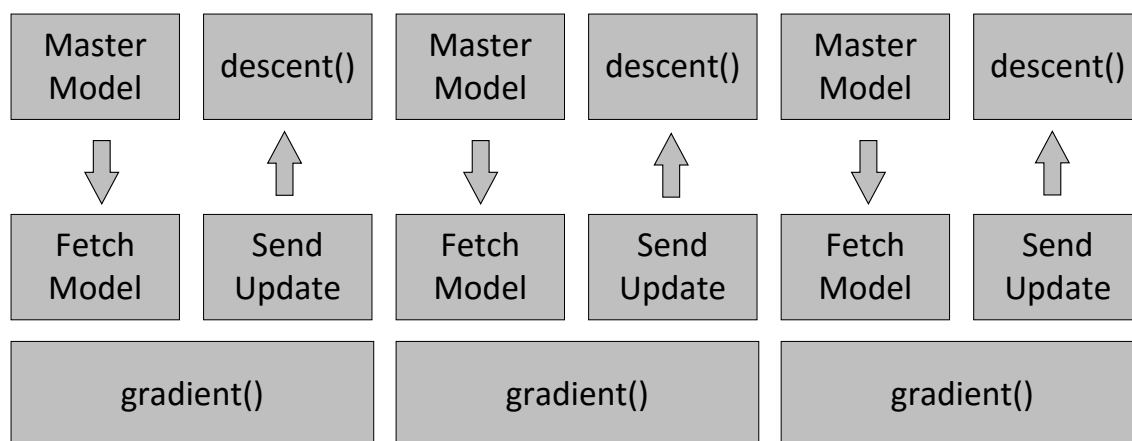


图 4-15 梯度下降并行化示意图

相较于 Spark MLIB 的执行过程，在计算梯度下降的过程中，其总是将所有的梯度计算计算完成后，才会执行对参数的更新（如下图），这样会导致 doc2vec 算法在训练过程中的平均速率下降。

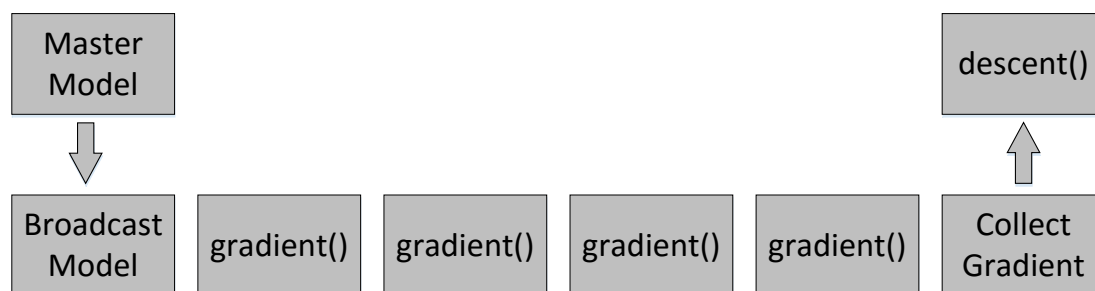


图 4-16 Spark 上梯度下降计算流程图

本算法是基于 Negative Sampling 的 CBOW 模型来优化 doc2vec 算法，Negative Sampling 错误!未找到引用源。是 Tomas Mikolov 用来提高 word2vec 训练速度并改善词向量质量，其利用随机负采样方式来代替原来的 Hierarchical Softmax 方式。CBOW 模型是在已知词的上下文 Context(w)的情况下,来预测 w 的概率（如下图所示），

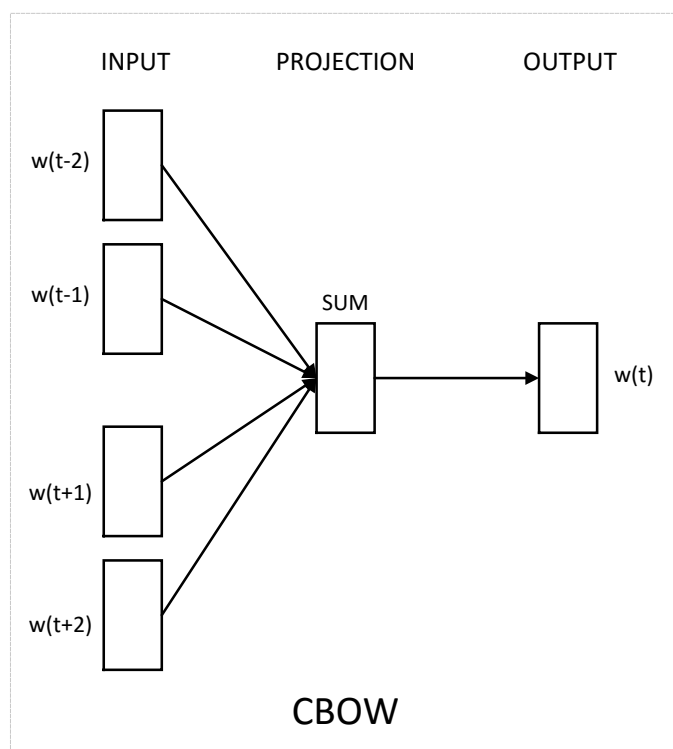


图 4-17 CBOW 模型示意图

基于 Negative Sampling 的采样方式认为词 w 本身为正采样，其他词为负采样。Word2vec 算法采用带权负采样算法来确定负样本子集，权重为单词在语料中的频率。采用 Negative Sampling 的 doc2vec 算法如下：

假设已经采集了关于 w 的负样本子集 NGE(w)。定义如下公式：

$$L^w(\tilde{w}) = \begin{cases} 1, & \tilde{w} = w; \\ 0, & \tilde{w} \neq w, \end{cases} \quad \forall \tilde{w} \in D$$

其中， $L^w(\tilde{w})$ 表示对词 \tilde{w} 的标签，即正样本为 1，负样本为 0。

对于正样本 (Context(w), w), 我们希望最大化如下概率:

$$g(w) = \prod_{u \in \{w\} \cup NEG(W)} P(u | Context(w))$$

其中:

$$p(u | Context(w)) = \begin{cases} \sigma(x_w^T \theta^u), L^w(u) = 1; \\ 1 - \sigma(x_w^T \theta^u), L^w(u) = 0, \end{cases}$$

$$(\sigma(x) = \frac{1}{1+e^{-x}} \text{ 为sigmoid函数})$$

写成整体表达式为:

$$p(u | Context(w)) = [\sigma(x_w^T \theta^u)]^{L^w(u)} \cdot [1 - \sigma(x_w^T \theta^u)]^{1-L^w(u)},$$

其中, x_w 表示 Context(w) 中各词向量之和, θ^u 为词 w 对应的辅助向量, 其为模型训练参数。将上述带入 $g(w)$ 可得:

$$g(w) = \sigma(x_w^T \theta^w) \prod_{u \in \{w\} \cup NEG(W)} [1 - \sigma(x_w^T \theta^u)]$$

其中 $\sigma(x_w^T \theta^w)$ 表示在上下文为 Context(w) 的情况下, 预测出单词 w 的概率, 而 $\sigma(x_w^T \theta^u)$ 表示为在上下文为 Context(w) 时, $u \in NEG(w)$ 的概率, 则上式最大化就是要使得在上下文为 Context(w) 的情况下, 增大正样本的概率同时降低负样本的概率。则对于给定的语料库 C, 函数

$$G = \prod_{w \in C} g(w)$$

可以作为需要优化的目标函数, 对 G 取对数就可以得到对数最大似然估计的目标函数如下:

$$\begin{aligned} \mathcal{L} &= \log G = \log \prod_{w \in C} g(w) = \sum_{w \in C} \log g(w) \\ &= \sum_{w \in C} \log \prod_{u \in \{w\} \cup NEG(W)} [\sigma(x_w^T \theta^u)]^{L^w(u)} \cdot [1 - \sigma(x_w^T \theta^u)]^{1-L^w(u)} \\ &= \sum_{w \in C} \sum_{u \in \{w\} \cup NEG(W)} \{L^w(u) \log[\sigma(x_w^T \theta^u)] + (1 - L^w(u)) \log[1 - \sigma(x_w^T \theta^u)]\} \end{aligned}$$

其中, x_w^T 和 θ^u 为需要训练的模型参数, 这里采用随机梯度下降算法来进行参数优化, 为了计算方便, 记:

$$\mathcal{L}(w, u) = L^w(u) \log[\sigma(x_w^T \theta^u)] + (1 - L^w(u)) \log[1 - \sigma(x_w^T \theta^u)]$$

通过随机梯度下降算法, 可以求得 $\mathcal{L}(w, u)$ 关于 θ^u 的梯度计算公式为:

$$\begin{aligned}
\frac{\partial \mathcal{L}(w, u)}{\partial \theta^u} &= \frac{\partial}{\partial \theta^u} \{L^w(u) \log[\sigma(x_w^T \theta^u)] + (1 - L^w(u)) \log[1 - \sigma(x_w^T \theta^u)]\} \\
&= L^w(u) [1 - \sigma(x_w^T \theta^u)] x_w - (1 - L^w(u)) \sigma(x_w^T \theta^u) x_w \\
&= \{L^w(u) [1 - \sigma(x_w^T \theta^u)] - (1 - L^w(u)) \sigma(x_w^T \theta^u)\} x_w \\
&= [L^w(u) - \sigma(x_w^T \theta^u)] x_w
\end{aligned}$$

可得, θ^u 的更新公式为:

$$\theta^u := \theta^u + \eta [L^w(u) - \sigma(x_w^T \theta^u)] x_w$$

同理, 可以求得 $\mathcal{L}(w, u)$ 关于 x_w 的梯度:

$$\frac{\partial \mathcal{L}(w, u)}{\partial x_w} = [L^w(u) - \sigma(x_w^T \theta^u)] \theta^u$$

于是, 利用上式可以求得 $v(\tilde{w})$, $\tilde{w} \in \text{Context}(w)$ 的更新公式为:

$$v(\tilde{w}) := v(\tilde{w}) + \eta \sum_{u \in \{w\} \cup \text{NEG}(W)} \frac{\partial \mathcal{L}(w, u)}{\partial x_w}$$

其中, 该公式表示对每个词向量 \tilde{w} 而言, 其由向量之和的分量求得。

则根据上式推倒, 可以得到基于 Negative Sampling 的 CBOW 模型采用随机梯度下降法更新各参数的伪代码如下:

开始: $e=0$, $x_w = \sum_{u \in \text{Context}(w)} v(u)$ 其中 x_w 为各词向量的加和, 词向量初始随机

```

1. FOR  $u \in \{w\} \cup \text{NEG}(W)$  DO
    {
         $q = \sigma(x_w^T \theta^u)$ 
         $g = \eta [L^w(u) - q]$ 
         $e := e + g \theta^u$ 
         $\theta^u := \theta^u + g x_w$ 
    }
2. FOR  $u \in \text{Context}(w)$  DO
    {
         $v(u) := v(u) + e$ 
    }

```

基于该算法, 本文提出了对其改进的并行化算法, 算法主要对求梯度下降过程中迭代计算的两个动态更新的参数进行并行化处理, 其中一个参数为 `syn0` 其代表最终得到词向量, 另一个参数是 `syn1neg` 其代表中间向量 θ^u , 通过在计算过程中缓存迭代生成的参数, 并在下一次迭代时更新现有的参数, 以这种方式并行化每次迭代过程, 并不断更新算法来完成算法的并行化, 其主要思路如下所示:

输入:

1. `model.syn0` 动态更新的参数, 其代表词向量 $v(x_w)$
2. `model.syn1neg` 动态更新的参数, 其代表中间向量 θ^u

在实现过程中, 将 `w` 按照 `NEG(w)` 进行分区, 并保存 `w` 的上下文 `Context(w)` 进行保存。

训练:

1. 分别对分区的 `NEG(w)` 和 `w` 进行训练, 获得 `syn0` 和 `syn1neg` 参数, 并保存到 Spark 的 RDD 中。
2. 对所有计算完毕的 `syn0` 和 `syn1neg` 的 RDD 调用 `RDD.aggregate` 方法合并成一个 RDD, 并进行缓存。
3. 将新生成的 RDD 训练参数广播给分区后的训练模型。
4. 将步骤 3 中训练好的参数合并到缓存的 RDD, 并进行迭代计算, 直到迭代结束。

4.4 本章小结

本章设计了基于电影评论的文本倾向分析系统, 首先, 本章给出了整个系统的设计框架图, 通过整体框架图可以知道, 文本倾向分析系统分为文本预处理模块、文本存储模块、文本分析模块, 之后详细阐述了各个模块的设计思路和关键技术。其次, 本章给出了基于 `doc2vec` 和 `LDA` 的评论文本倾向分析算法, 文本倾向分析其实质是对文本的二分类, 文本分类问题最核心的内容是如何提取文本特征, 基于对文本特征提取的研究, 本章设计了融合段向量和主题模型的文本特征融合算法, 段向量采用 `Doc2vec` 算法获取, 其标识了文本段的相似特性, 主题向量采用 `LDA` 算法获取, 其标识了文本段的主题特性, 通过结合这两个特性可以很好的反映一段评论的倾向性(主题倾向和情感倾向), 然后再训练分类模型可以得到理想的效果。最后, 为了提高算法的训练速度, 本章讨论了如何借助 Spark 大数据平台对算法进行并行化处理, 通过对分词算法、`LDA` 算法、`Doc2vec` 算法分别进行并行化处理可以大大提高算法的训练速度, 提升整个系统的效率。通过本章的介绍, 完成了对基于大数据平台的, 针对中文电影评论倾向分析的文本分析

系统的设计，之后可以根据设计完成整个系统的实现。

第五章 基于大数据平台的文本倾向分析系统实现及验证测试

本章是在第 4 章的基础上对基于电影评论的文本倾向分析系统的原型系统实现以及验证测试。首先本章针对系统设计阶段中提出的三个主要模块进行具体实现，然后根据具体搭建整个原型系统，最后通过验证测试给出系统的测试结果。系统搭建主要介绍的是各模块的部署以及核心代码展示，通过对文本预处理模块，文本存储模块，文本分析模块三个核心模块的整合，完成整个原型系统的搭建。系统的验证测试主要通过两个方面来测试系统的各项指标：1.准确率测试，2.速度测试，通过这两个指标来确定整个系统的综合测试结果。准确率测试采用 ROC 测试方式来测试文本倾向分析算法的准确性，速度测试方面采用速度测试脚本来测试整个系统在模型训练方面的速度性能。

5.1 系统各模块的实现

5.1.1 文本预处理模块实现

文本预处理模块是针对豆瓣电影网站用于摄取评论语料并对语料进行自动标注的模块，该模块主要由两部分组成：1.针对豆瓣电影网站的定制化爬虫模块，2.基于豆瓣电影评论的自动标注模块，本节是对这两个子模块的具体实现进行阐述。

(1).定制化爬虫模块

针对豆瓣电影网站的定制化爬虫是以 Scrapy 框架为基础，通过设计特定的摄取策略来对豆瓣电影网站进行爬取，基于 Scrapy 的爬虫系统是通过 spider 模块、pipelines 模块、item 模块构成的爬虫系统，不同的模块负责爬虫不同的处理任务，爬虫的程序流程图如下所示：

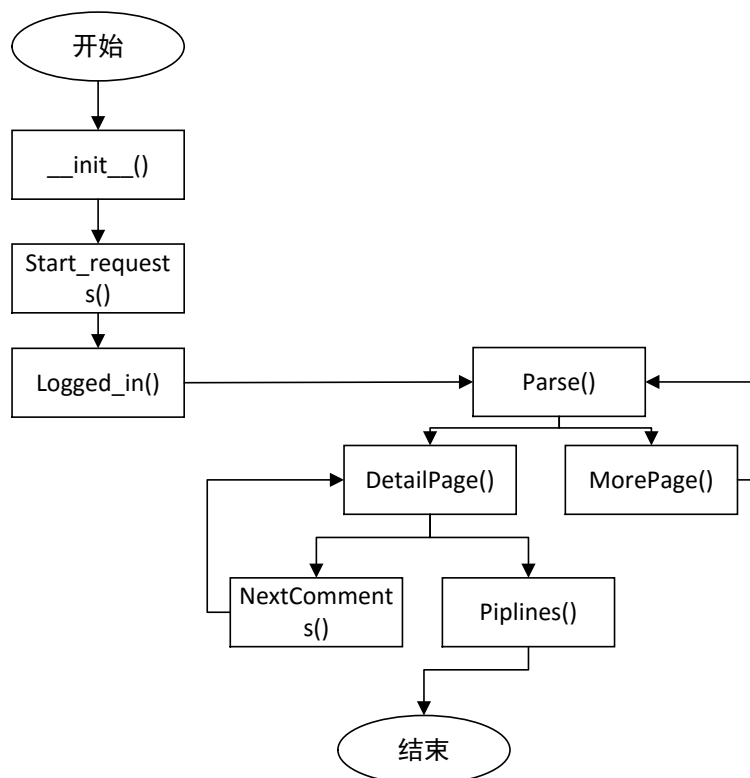


图 5-1 定制化爬虫程序流程图

首先，爬虫通过 `_init_` 构造函数进行初始化，该函数主要是用于初始化 `start_urls` 列表，`start_urls` 列表里保存了需要爬取的种子入口，该入口即为豆瓣电影的登录页面，`start_requests()` 函数会根据 `start_urls` 列表里的 URL 发送 `request` 请求，该请求可以自定义需要发送的 `http` 头参数，在这里可以带上用户名和密码信息，用来模拟登录功能，登录后会调用回调函数 `logged_in()` 函数，`logged_in` 函数的参数是 `response` 类型的对象，该对象包含了需要分析的首页 `html` 数据，通过调用 `parse()` 函数来分析首页 `html` 数据，这里用来获取热门电影的详情页 `url` 和“加载更多”按钮的 `url`，分析出详情页的 `url` 后构造 `scrapy.request` 对象并发送 `request` 请求，该 `request` 请求的回调函数为 `DetailPage()` 函数，该函数用来分析详情页中的评论信息，将评论信息主体、打分、评论点赞数封装为 `items()` 对象并发送给 `pipelines()` 函数，同时，分析翻页链接并构造下一页链接对应的 `scrapy.request` 对象，该对象的回调函数同样是 `DetailPage()`。`Pipelines` 函数将获取到的评论信息 `item` 缓存到文件中，并以电影名命名该文件，当电影的评论信息被爬取完毕后，爬虫结束。

在摄取过程中，等爬虫递归到一定深度时，豆瓣网站会跳转到登陆页，要求用户进行登录验证，所以爬虫需要实现模拟登陆功能才能摄取到足够的评论文本，模拟登陆功能主要是通过模拟登陆 `post` 请求，提交 `post` 表单，然后保存 `response` 返回的 `cookie` 信息，之后在所有请求中跟踪 `cookie` 信息达到保持爬虫和豆瓣网站之间状态的目的，这样在摄取过程中，爬虫不会因为摄取深度过深而被要求登陆。除了需要在 `HTTP` 请求中

post 正确的表单，还需要爬虫模拟 http 的请求头来防止目标网站拒绝服务，因为许多网站通过判断 http 请求头的方式来过滤爬虫，本系统设计的爬虫 http 头信息如下表所示：

表 5-1 爬虫模拟 http 头信息

Key	Value
User-Agent	Mozilla/5.0(Windows NT 10.0;Win64;x64;rv:49.0) Gecko/20100101 Firefox/49.0
Accept	*/*
Accept-Language	Zh-ch,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding	gzip,deflate,br
Connection	keep-alive
Cookie	bid=aASBX_5Vb8W;ps=y
Upgrade-Insecure-Requests	1

同时，在模拟登陆时，如果爬虫频繁的请求登陆页，会触发网站的验证码机制，这是大多数登陆网站设计的用来防止恶意登陆的措施，在遇到这种情况时，可以通过判断登录页面是否需要输入验证码，如果需要填写验证码，就将验证码图片保存下来，然后人工识别后，在连同用户名和密码一起发送请求，完成登陆验证，整个模拟登陆的流程如图如下所示：

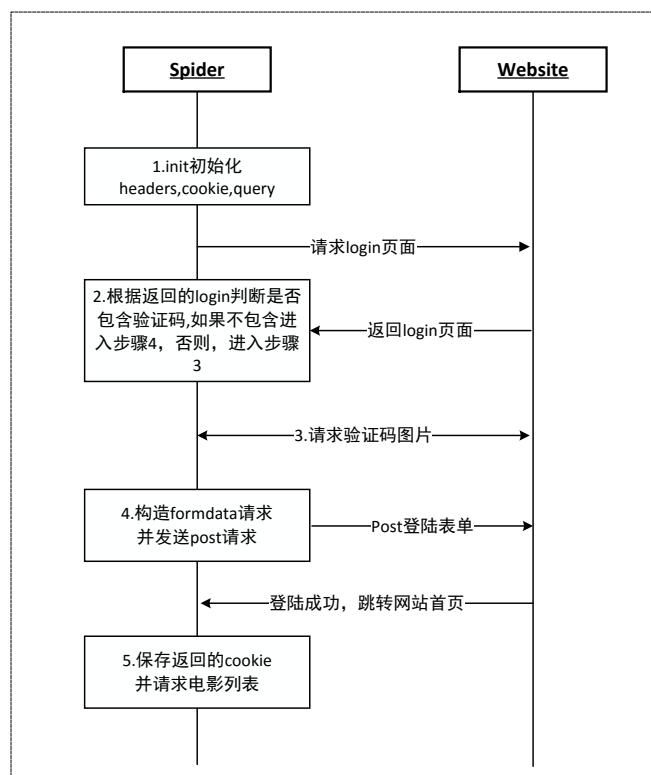


图 5-2 模拟登陆流程示意图

(2).自动化标注模块

自动化标注模块是对爬取到的评论进行正负性标注，标注后的评论文本可以作为训练语料进行训练，从而可以获取到正负文本的特征，由于采用人工标注的形式需要大量的人力来完成，虽然标注的准确度比自动标注高，但是效率极低，采用自动化标注的方式可以提高标注效率，但是如果标注策略不当，会导致标注不准确，这样的语料训练生成的模型也是不准确的，从而不能达到满足要求的准确率，所以需要优化标注策略来提高标注的准确度，本系统采用的标注策略是基于情感词典和标签融合的标注算法，标签可以近似为人工标注的结果，但是单单基于标签不能准确的标注评论的倾向性，所以需要用情感词典进行优化，使用情感词的正负级性来计算整个句子的情感值，从而可以为句子标注提供参考系数优化标注结果。下图是基于以上思路的自动化标注模块的程序流程图：

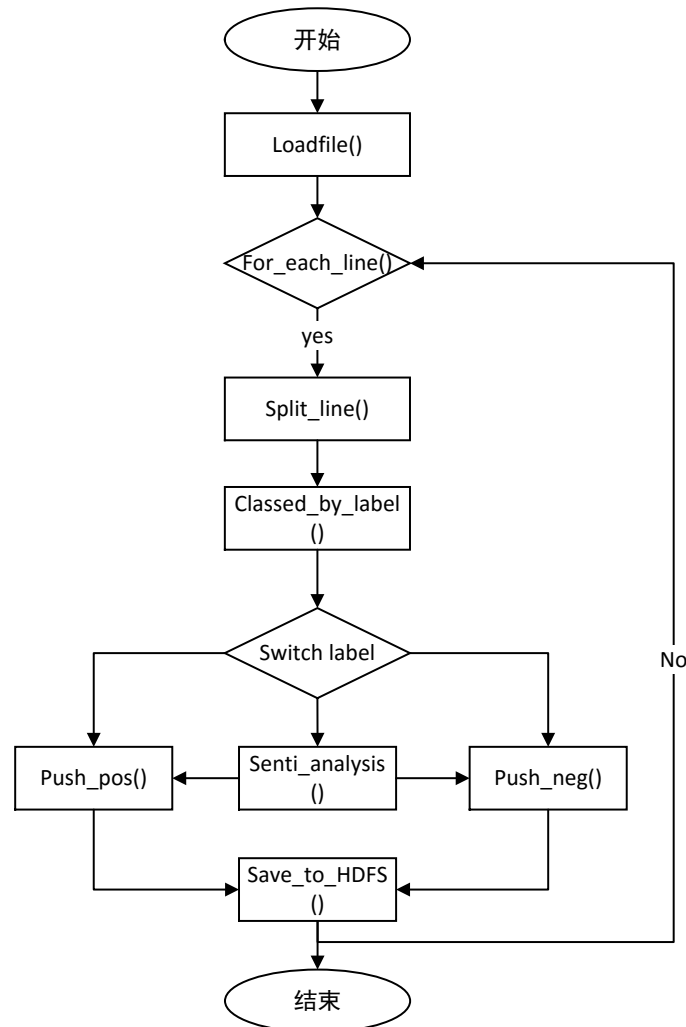


图 5-3 自动化标注模块程序流程图

首先，自动标注模块通过 `loadfile()` 载入评论文本，然后通过 `for_each_line()` 对评论文本按行进行遍历，遍历的每行都是一个带标注的评论文本，之后，通过切词程序对文本

进行切词处理，并通过 `classed_by_label()` 函数获取到该文本的标签（这里的切词过程可以先通过并行化的切词处理后在进行标注）。然后，通过文本标签对文本进行分类，如果标签值大于某个正阈值就将其分为正类，如果标签值小于某个负阈值就将其分为负类，如果标签值介于正阈值与负阈值之间，则通过 `senti_analysis()` 模块分析该文本的实际标签值，`senti_analysis()` 函数通过情感词典计算出一个权重，并计算加权后的标签值，然后通过该标签值将该评论文本分为正类或负类，通过这种策略遍历所有行后，标注模块结束。

5.1.2 文本存储模块实现

文本存储模块主要是实现对语料库的存储和对评论文本的存储，系统采用 HDFS 作为存储文本的文件系统，通过 HDFS 连接文本预处理模块和文本分析模块，文本预处理模块将下载的评价信息按照电影名分类保存到文件系统上，同时在数据库中保存文件的元信息，然后通过标注模块调用封装好的 HDFS 接口，对评论文本进行标注生成语料库，文本分析模块通过读取 HDFS 上的语料库来训练模型，文本存储模块封装了 HDFS 调用的接口，对于文本预处理模块和文本分析模块而言，其是透明的，只需要通过相应的接口来上传和下载文件即可。

本文通过封装 HDFS 常用接口来提供对 HDFS 的操作，包括 `makedirs()`，`create()`，`copyFromLocal()`，`exists()`，`delete()`，`fileStatus()` 等操作，其中，`makedirs()` 为创建目录命令，其参数为目录路径；`create()` 为创建文件，其参数为文件路径；`copyFromLocal()` 为将本地文件拷贝到 HDFS 文件系统；`exists()` 为查看文件或目录是否存在；`delete()` 为永久的删除指定的文件或目录；`fileStatus()` 返回一个文件或目录的元信息对象，用来查询文件或目录的元信息。

5.1.3 文本分析模块实现

文本分析模块是对文本进行特征抽取、特征融合，然后通过训练特征向量来完成文本分类任务，文本分析模块的实现是通过运行在 Spark 上的 Driver Application 完成的，其中特征抽取功能需要实现对 LDA 模型和 doc2vec 模型的训练，特征融合功能需要对 LDA 特征向量和 doc2vec 特征向量进行融合形成新的特征向量，然后通过 SGD 分类器进行分类模型训练，整个模块的 Spark 程序流转图如下所示：

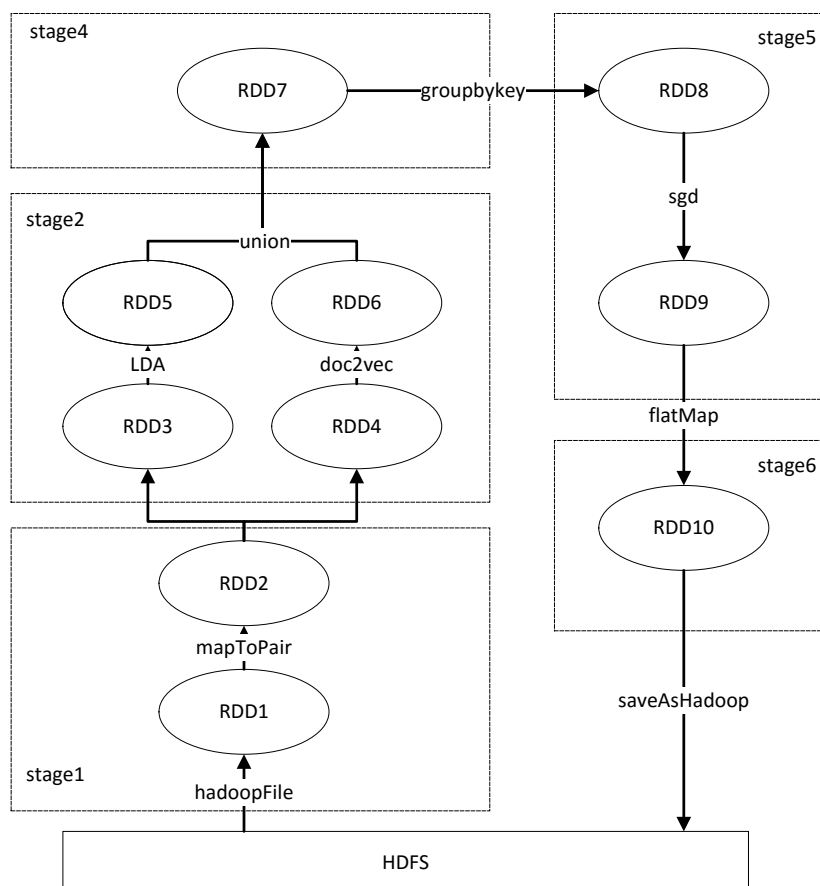


图 5-4 文本分析模块 Spark 程序流转图

如上图所示，文本分析模块的 Spark 程序流转图，首先，需要从 HDFS 中获取用于训练的语料并将其转换为 RDD1，然后通过 `mapToPair` 算子将语料按标签映射为 RDD2，并 `copy` 两份生成 RDD3 和 RDD4，通过 LDA 算法和 `doc2vec` 算法分别提取出语料的特征向量生成 RDD5 和 RDD6，然后通过特征融合算法生成 RDD7，之后通过 `groupByKey` 算子生成待分类的文本特征向量 RDD8，通过 `SGDClassifier` 训练文本分类模型生成 RDD9，然后使用 `flatMap` 算子将其映射为 RDD10，并使用 `saveAsHadoop` 方法将文本分类模型持久化到 HDFS 上。整个文本训练过程到此结束。

在模型训练过程中，最主要的是训练 LDA 模型和 Doc2vec 模型，这两个模型是用来完成文本特征提取的关键，LDA 模型的训练与 Doc2vec 模型的训练过程有相似处，也有不同点，以下通过两个模型的训练程序流程图来对比两个模型的特点，并给出融合 lda 模型和 doc2vec 模型的算法流程图。

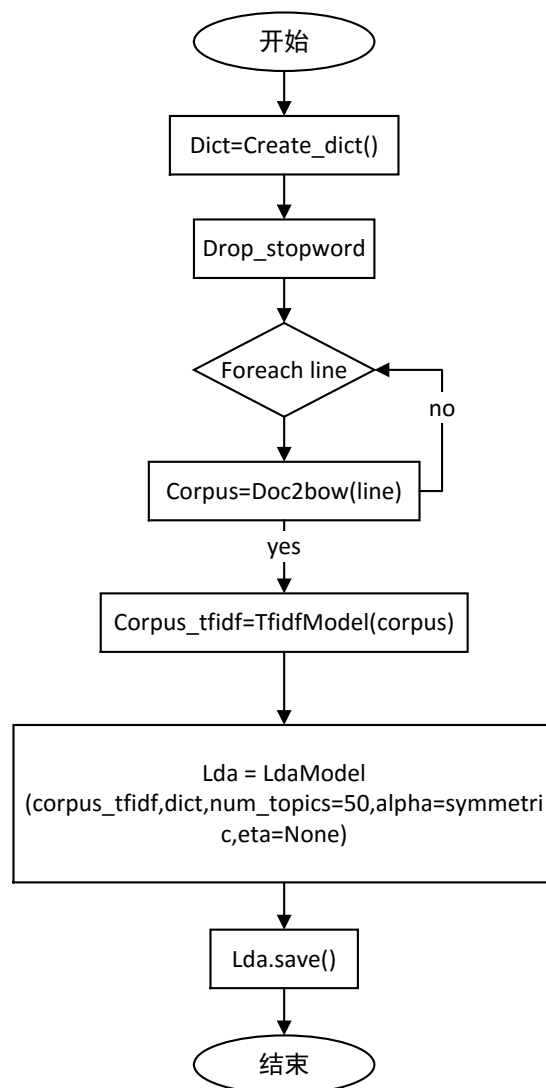


图 5-5 LDA 模型训练程序流程图

如上图所示，LDA 模型的训练是建立在 BOW(Bag of Words)模型的基础上的，所以首先需要建立语料词典，通过建立语料词典并去除停用词后，构造 BOW(Bag of Words)语料。词袋语料将词转换为词典下标和词频，例如如下的语料：

["Human machine interface for lab abc computer applications",
 "A survey of user opinion of computer system response time",
 "The EPS user interface management system",
 "System and human system engineering testing of EPS",
 "Relation of user perceived response time to error measurement"]

经过转换后，会转换成如下的格式：

[(0, 1), (1, 1), (2, 1)]
 [(0, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)]
 [(2, 1), (5, 1), (7, 1), (8, 1)]

[(1, 1), (5, 2), (8, 1)]

[(3, 1), (6, 1), (7, 1)]

其中每个 list 表示一条文本(或文档、语料), 每个 list 中是一个二元组 (word_id, word_count), 表示单词在词典中的索引 ID 和单词在整个语料库中出现的词频。通过这种方式转换的语料库可以利用 TF-IDF 将每个单词归一化为 0-1 的 float 型数字, 这样便于 LDA 模型的训练, 之后将归一化后的语料作为输入来训练 LDA 模型, LDA 算法的参数包括 6 个主要参数, 第一和第二个参数分别为语料和词典, 第三个参数为模型训练的主题个数(一般主题数多于实际主题个数), 第四和第五个参数分别是 LDA 模型中的重要参数, 其作为影响文档-主题分布 θ 和主题-词分布 λ 的超参数, 在默认情况下, 其被设置为 $1/\text{num_topics}$ 。当训练完成后, 我们可以通过 save() 函数将模型保存起来。

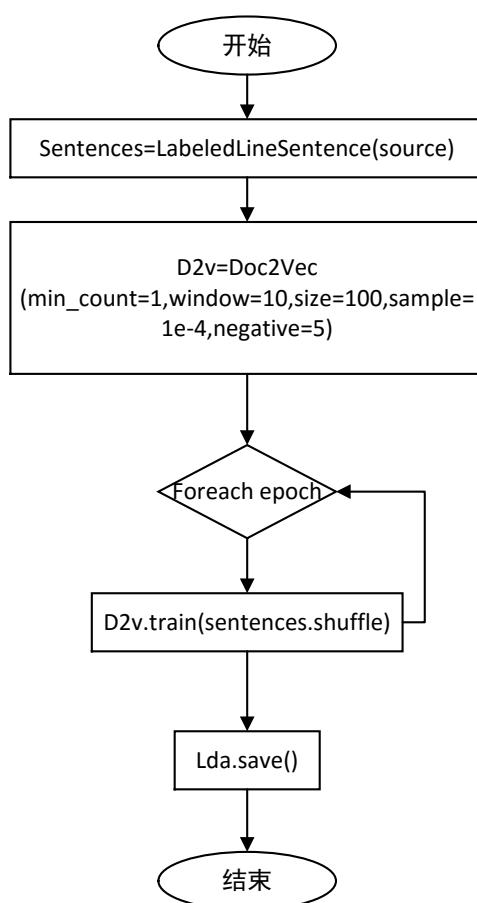


图 5-6 Doc2vec 模型训练程序流程图

上图是 Doc2vec 模型训练的程序流程图, 其与 LDA 模型的区别是不需要对语料进行特殊处理, 只需要为每个文本(文档、段落、句子)做一个唯一的标记, 该标记通过 LabeledLineSentence() 函数实现, 其目的是为了之后迭代训练过程中可以区分不同的文本, 之后就可以通过 Doc2vec 方法构造 doc2vec 模型, doc2vec 模型也需要配置相应的参数, 其中 min_count 表示忽略词频小于 min_count 的单词, window 表示当前单词的上下文单词个数, size 表示训练词向量的维度, samples 表示配置高频词的随机采样阈值,

negative 表示是否启用负采样算法，当 **negative** 的值>0 时启用，其值表示了多少个“噪声单词”需要考虑，一般取值为 5-20。同样，我们可以用 **save()**方法保存训练好的模型。

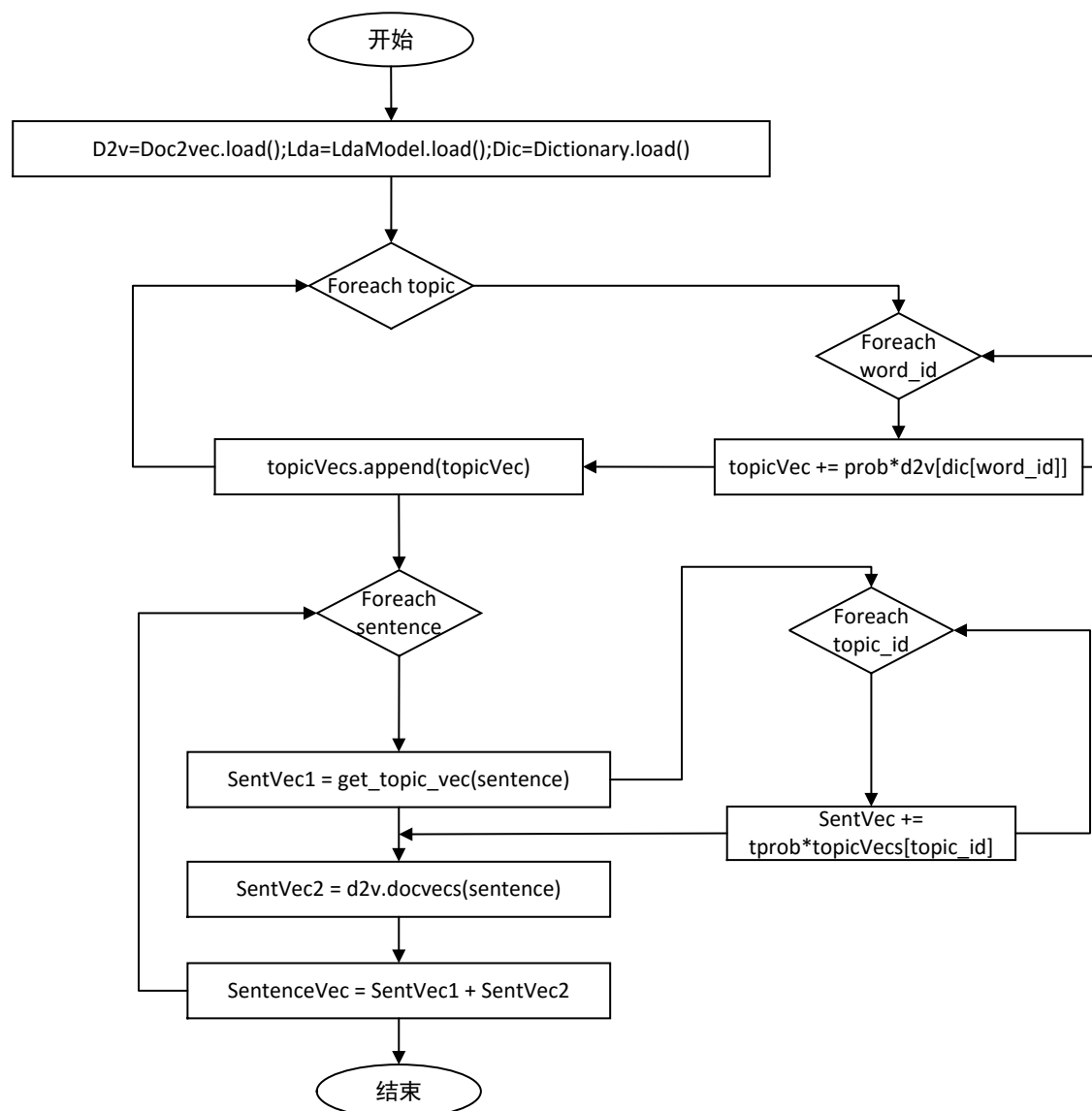


图 5-7 融合 LDA 的 Doc2vec 算法实现程序流程图

训练好 LDA 模型和 Doc2vec 模型后，通过利用融合 LDA 的 Doc2vec 算法来对文本进行特征向量表示，该算法的程序流程图如上图所示，首先，需要将 LDA 模型和 Doc2vec 模型导入到内存中，这些模型是事先训练好的。然后遍历 LDA 中的主题分布，并计算主题分布中每个单词的词向量与概率的乘积，将主题中所有词的词向量进行加和就可以得到主题向量 **topicVec**，然后将所有主题向量 **topicVec** 保存到 **topicVecs** 数组中，之后遍历整个语料库，对每个句子的文档-主题分布，根据 **topicVecs** 计算句子的 **sentVec1**，然后再通过 **doc2vec** 模型获取该句子的 **sentVec2**，之后将这两个向量进行叠加，得到最终的句子向量 **sentVec**，当遍历了语料库中所有的句子后，就可以得到语料库的特征向量矩阵 **sentencesVecs**。之后，可以利用分类模型，来验证文本语料的特征向量是否准确，同

时，我们可以利用该特征向量矩阵对句子进行倾向性分析。

5.2 系统环境搭建与系统部署

本系统采用 Spark 集群作为模型训练平台，使用 python 作为原型系统开发语言，整个系统的部署架构图如下所示：

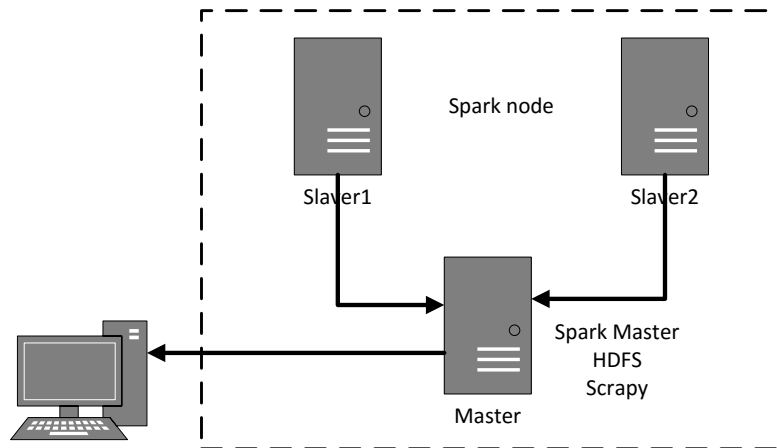


图 5-8 系统部署架构图

系统采用混合部署，其中 spark 集群采用一个 master 结点和两个 slaver 结点构成，master 结点上同时部署 HDFS 和 Scrapy 爬虫。Master 作为整个系统的调度中心，用来集成系统的三个模块，同时，Master 还是 Spark 任务的提交结点，测试程序主要运行在 Master 结点上，包括功能测试、准确率测试、性能测试。

原型系统中硬件环境及软件配置如下表：

表 5-2 系统软硬件环境配置表

序号	设备名称	硬件	软件	数量
1	Spark Node1	CPU: Inter(R) Core(TM) i3 CPU T 1350 @ 1.86GHz; RAM: 8G; 硬盘: 250G; 网卡: 三块	操作系统: Fedora Linux version 2.6.28 平台: Spark v1.6.1 (JVM version 7 Python version 7)	1
2	Spark Node2	CPU: Inter(R) Core(TM) i5- 3470 CPU @ 3.20GHz; RAM: 8G; 硬盘: 250G;	操作系统: Fedora Linux version 2.6.28 平台: Spark v1.6.1 (JVM version 7	1

		网卡：三块；	Python version 7)	
3	Spark Master	CPU: Inter(R) Core(TM) i3 CPU T 1350 @ 1.86GHz; RAM: 8G; 硬盘: 500G; 网卡: 三块	操作系统: Fedora Linux version 2.6.28 平台: Spark v1.6.1 文件系统: HDFS v2.7.2 (JVM version 7 Python version 7)	1

本系统采用 Scrapy 作为文本爬取的爬虫框架，所以需要在 master 结点安装 Scrapy，Scrapy 是用 Python 开发的爬虫框架，根据需求进行二次开发，使用 Python 的 virtualenv 环境安装 scrapy，防止其与系统的 Python 版本发生冲突，安装 virtualenv 并安装 Scrapy 的步骤如下所示：

首先，安装 pip，然后用 pip 源安装 virtualenv

```
#yum install python-setuptools python-devel
```

```
#easy_install virtualenv
```

```
#easy_install pip
```

```
#pip install virtualenv
```

其次，安装好 virtualenv 后使用 virtualenv 构建 scrapy 所需的 Python 环境

```
#virtualenv scrapy-env
```

```
#virtualenv --python=/usr/local/python-2.7.8/bin/python2.7 scrapy-env
```

```
#virtualenv --system-site-package scrapy-env
```

最后，进入虚拟目录，启动虚拟环境并安装 scrapy

```
#cd scrapy-env
```

```
#source bin/activate
```

```
#pip install scrapy
```

同时，系统需要搭建 Spark 集群，Spark 安装依赖于 jdk，所以要求系统环境必须安装好 jdk，然后从官网(<http://spark.apache.org/downloads.html>)下载合适的 Spark 安装包，本系统选择的是 Pre-built for Hadoop 2.7 版本的 Spark 程序，以下是对 Spark 集群的搭建过程：

```
#tar -zxvf spark-2.0.1-bin-hadoop2.7
```

```
#cd spark-2.0.1-bin-hadoop2.7/conf
```

```
#cp spark-env.sh.template spark-env.sh
```

修改 spark-env.sh，在文件末尾导入 JAVA_HOME

```
export JAVA_HOME=/usr/local/jdk1.6.0_24/
```

修改配置文件 `slaves`，添加 `slaves` 节点

```
#echo slaver1 >> slavers
```

```
#echo slaver3 >> slavers
```

配置好 `master` 节点后，把 `master` 上 `Spark` 软件包 `copy` 到 `slaver1`、`slaver2` 的相同目录下，并在 `Master` 节点上运行命令：

```
#cd $SPARK_HOME/bin
```

```
#./start-all.sh
```

至此，系统所需的运行环境已经搭建完毕，之后需要部署系统所需的各个模块，系统部署的整个流程图如下所示：

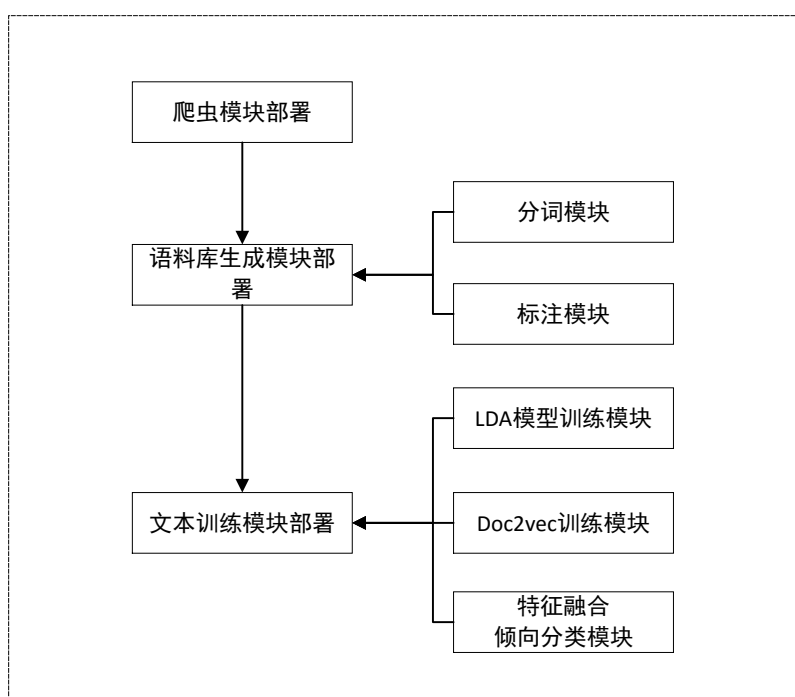


图 5-9 系统部署流程图

如上图所示，首先需要部署的是爬虫模块，爬虫模块是基于 `Scrapy` 构造的定制化爬虫工程，其包含了爬取豆瓣电影所需要的各个模块，然后需要部署的是语料库生成模块，语料库生成模块包括分词模块和标注模块，分词模块是基于 `jieba` 分词实现的分词模块，为了提高分词速度最后需要部署文本训练模块，文本训练模块包括 `LDA` 模型训练模块，`doc2vec` 模型训练模块和特征融合倾向分析模块。

5.3 核心功能模块展示

5.3.1 爬虫核心模块

爬虫基于 Scrapy 框架编写，首先需要建立爬虫工程：

```
#scrapy startproject douban
```

这时会创建爬虫的工程目录如下：

```
├── douban
│   ├── __init__.py
│   ├── items.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       ├── __init__.py
└── scrapy.cfg
```

这些文件分别是：

scrapy.cfg: 项目的配置文件

douban/: 项目的工程文件，代码主要集中在这个文件夹下

douban/items.py: 项目的 item 文件

douban/pipelines.py: 项目的 pipelines 文件

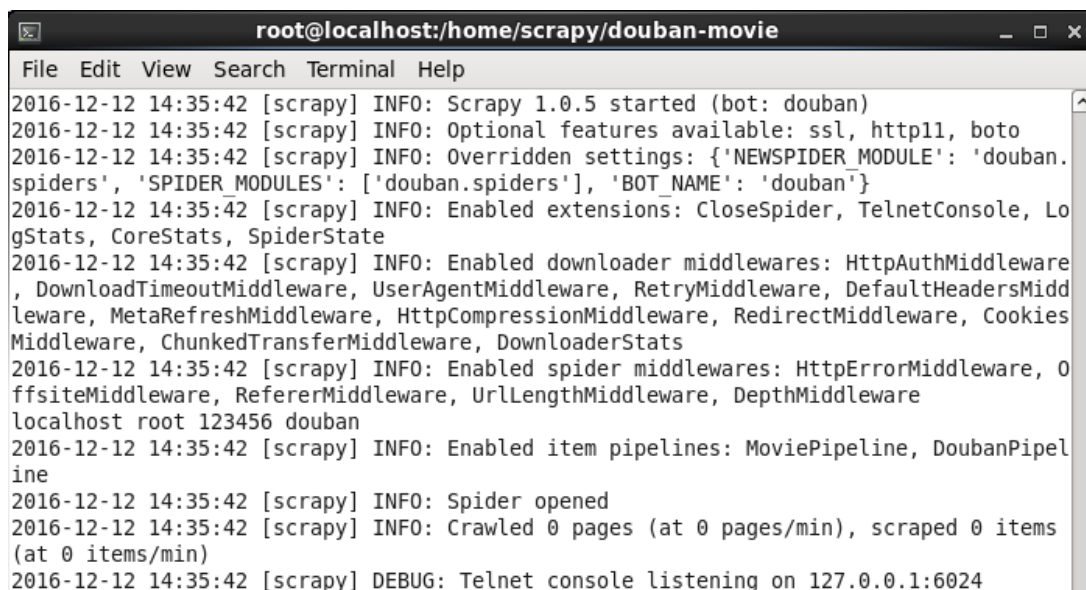
douban/setting.py: 项目的设置文件

douban/spiders/: 爬虫工程目录，这里实现爬虫的主要功能

爬虫模块主要实现对 douban 网站的电影评论进行爬取，其主要基于 Scrapy 构建，核心的内容包括配置文件、spider、pipelines，其中，配置文件配置了爬虫的基本参数，包括 HTTP HEADERS、表单信息 FORMDICT、请求 QUERY，然后如下命令启动爬虫：

```
#scrapy crawl douban
```

爬虫正常启动后会显示如下内容：



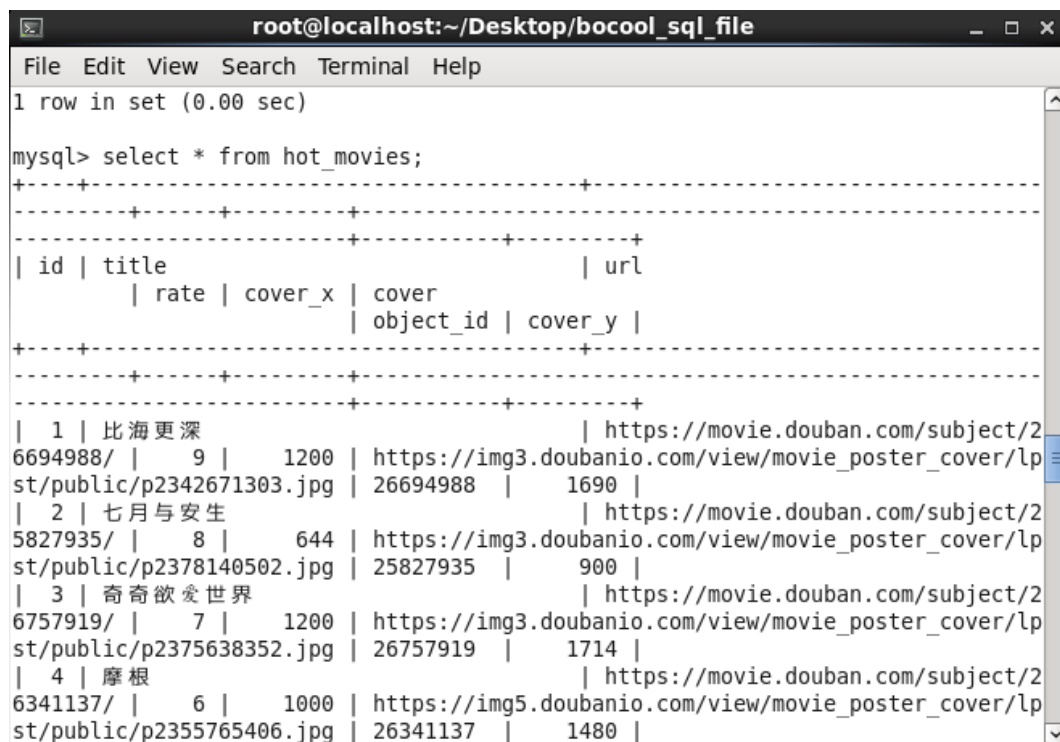
```

root@localhost:/home/scrapy/douban-movie
File Edit View Search Terminal Help
2016-12-12 14:35:42 [scrapy] INFO: Scrapy 1.0.5 started (bot: douban)
2016-12-12 14:35:42 [scrapy] INFO: Optional features available: ssl, http11, boto
2016-12-12 14:35:42 [scrapy] INFO: Overridden settings: {'NEWSPIDER_MODULE': 'douban.spiders', 'SPIDER_MODULES': ['douban.spiders'], 'BOT_NAME': 'douban'}
2016-12-12 14:35:42 [scrapy] INFO: Enabled extensions: CloseSpider, TelnetConsole, LogStats, CoreStats, SpiderState
2016-12-12 14:35:42 [scrapy] INFO: Enabled downloader middlewares: HttpAuthMiddleware, DownloadTimeoutMiddleware, UserAgentMiddleware, RetryMiddleware, DefaultHeadersMiddleware, MetaRefreshMiddleware, HttpCompressionMiddleware, RedirectMiddleware, CookiesMiddleware, ChunkedTransferMiddleware, DownloaderStats
2016-12-12 14:35:42 [scrapy] INFO: Enabled spider middlewares: HttpErrorMiddleware, OffsiteMiddleware, RefererMiddleware, UrlLengthMiddleware, DepthMiddleware
localhost root 123456 douban
2016-12-12 14:35:42 [scrapy] INFO: Enabled item pipelines: MoviePipeline, DoubanPipeline
2016-12-12 14:35:42 [scrapy] INFO: Spider opened
2016-12-12 14:35:42 [scrapy] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2016-12-12 14:35:42 [scrapy] DEBUG: Telnet console listening on 127.0.0.1:6024

```

图 5-10 爬虫启动过程

爬虫启动后会将爬取到的电影列表保存到 mysql 中，电影列表包含了电影的 subjectID，该信息是用来爬取评论详情页的重要信息，同时，电影列表页也可用来去重，防止重复爬取电影评论，爬取到的电影列表如下所示：



```

root@localhost:~/Desktop/bocool_sql_file
File Edit View Search Terminal Help
1 row in set (0.00 sec)

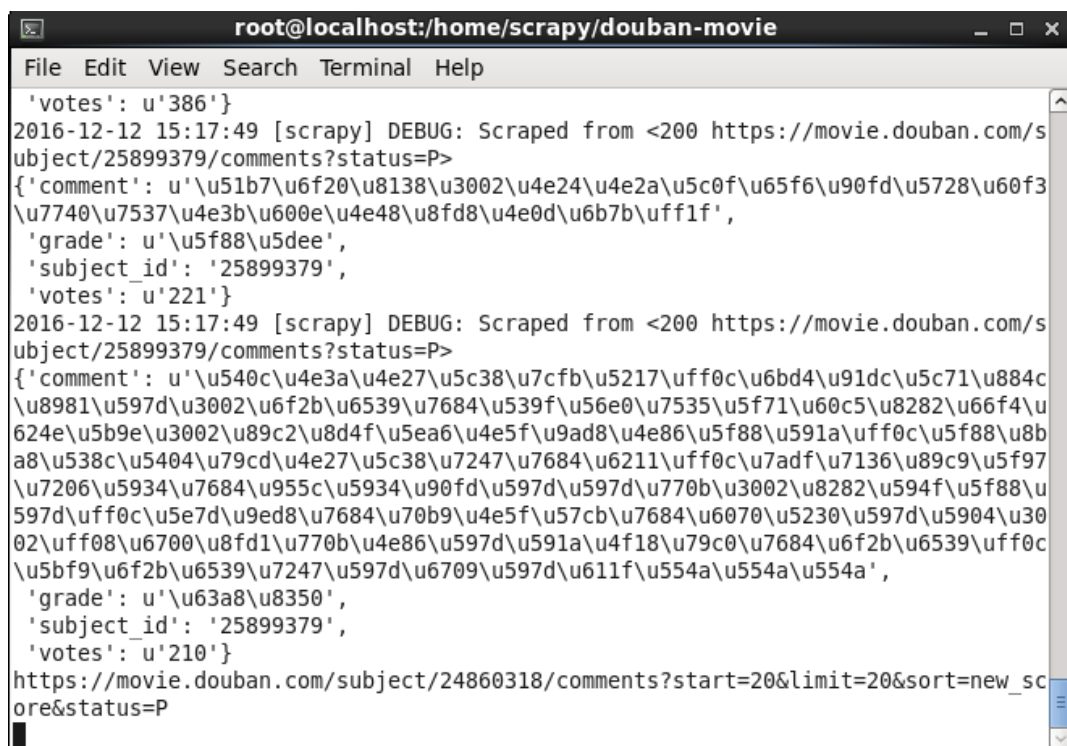
mysql> select * from hot_movies;
+-----+-----+-----+-----+-----+-----+-----+
| id | title | rate | cover_x | cover | object_id | cover_y | url |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 比海更深 | 9 | 1200 | https://img3.doubanio.com/view/movie_poster_cover/lps/public/p2342671303.jpg | 26694988 | 1690 | https://movie.douban.com/subject/26694988/ |
| 2 | 七月与安生 | 8 | 644 | https://img3.doubanio.com/view/movie_poster_cover/lps/public/p2378140502.jpg | 25827935 | 900 | https://movie.douban.com/subject/25827935/ |
| 3 | 奇奇欲爱世界 | 7 | 1200 | https://img3.doubanio.com/view/movie_poster_cover/lps/public/p2375638352.jpg | 26757919 | 1714 | https://movie.douban.com/subject/26757919/ |
| 4 | 摩根 | 6 | 1000 | https://img5.doubanio.com/view/movie_poster_cover/lps/public/p2355765406.jpg | 26341137 | 1480 | https://movie.douban.com/subject/26341137/ |
+-----+-----+-----+-----+-----+-----+-----+

```

图 5-11 豆瓣热门电影列表

通过热门电影列表，可以获取到豆瓣的电影评论详情页，然后通过 xpath 获取到电影的评论信息作为训练的语料，本系统采用爬虫爬取电影评论信息只是为了作为文本倾

向分析原型系统的初始语料，并不会恶意的去爬取豆瓣电影网站，获取到的电影评论也仅作为科研实验数据，以下是爬虫爬取电影评论时的运行状态：



```

root@localhost:/home/scrapy/douban-movie
File Edit View Search Terminal Help
'votes': u'386'}
2016-12-12 15:17:49 [scrapy] DEBUG: Scraped from <200 https://movie.douban.com/subject/25899379/comments?status=P>
{'comment': u'\u51b7\u6f20\u8138\u3002\u4e24\u4e2a\u5c0f\u65f6\u90fd\u5728\u60f3\u7740\u7537\u4e3b\u600e\u4e48\u8fd8\u4e0d\u6b7b\u5f1f',
'grade': u'\u5f88\u5dee',
'subject_id': '25899379',
'votes': u'221'}
2016-12-12 15:17:49 [scrapy] DEBUG: Scraped from <200 https://movie.douban.com/subject/25899379/comments?status=P>
{'comment': u'\u540c\u4e3a\u4e27\u5c38\u7cfb\u5217\u5f0c\u6bd4\u91dc\u5c71\u884c\u8981\u597d\u3002\u6f2b\u6539\u7684\u539f\u56e0\u7535\u5f71\u60c5\u8282\u66f4\u624e\u5b9e\u3002\u89c2\u8d4f\u5ea6\u4e5f\u9ad8\u4e86\u5f88\u591a\u5f88\u8ba8\u538c\u5404\u79cd\u4e27\u5c38\u7247\u7684\u6211\u5f0c\u7adf\u7136\u89c9\u5f97\u7206\u5934\u7684\u955c\u5934\u90fd\u597d\u597d\u770b\u3002\u8282\u594f\u5f88\u597d\u5f0c\u5e7d\u9ed8\u7684\u70b9\u4e5f\u57cb\u7684\u6070\u5230\u597d\u5904\u3002\u5f08\u6700\u8fd1\u770b\u4e86\u597d\u591a\u4f18\u79c0\u7684\u6f2b\u6539\u5f0c\u5bf9\u6f2b\u6539\u7247\u597d\u6709\u597d\u611f\u554a\u554a\u554a',
'grade': u'\u63a8\u8350',
'subject_id': '25899379',
'votes': u'210'}
https://movie.douban.com/subject/24860318/comments?start=20&limit=20&sort=new_score&status=P

```

图 5-12 爬虫爬取热门电影评论

5.3.2 语料生成模块

语料生成模块是将爬取的语料进行切词和标注，并通过编写 Spark Application Driver 来并行化的处理切词任务，切词任务需要将文本切分成单词，同时，还需要去掉标点符号等影响模型训练的噪声，生成的训练语料用于模型训练，包括主题模型训练，doc2vec 模型训练。

经过切词处理后，语料会被标注为训练语料、测试语料，其中训练语料中包含正向语料、负向语料、不确定语料三部分，测试语料分为正向测试语料和负向测试语料，语料的组织格式按照之前定义好的语料组织格式，保存到 HDFS 上，以下是经过处理和标注后的语料格式：



图 5-13 经切词和标注处理后的电影评论语料

5.3.3 文本训练模块

文本训练模块是对文本模型的训练过程，文本训练包括 lda 模型训练，doc2vec 模块训练，以及融合 lda 和 doc2vec 向量的模型训练。模型在训练过程中使用 Spark 集群并行化训练任务，同时将训练得到的模型持久化到 HDFS 上，以供之后其他模型训练时使用，文本训练模块的核心代码如下：

```
--
64 def lda_train(path):
65
66     corpus = MyCorpus(path)
67     dic = corpus.dictionary
68     #for vector in corpus:
69     #     print vector
70     corpora.MmCorpus.serialize('./douban.mm', corpus)
71
72     #序列化语料
73     corpus = corpora.MmCorpus('./douban.mm')
74     tfidf = models.TfidfModel(corpus)
75
76     corpus_tfidf = tfidf[corpus]
77     #for doc in corpus_tfidf:
78     #     print doc
79
80     lda = models.LdaModel(corpus_tfidf, id2word=dic, num_topics=100, \
81                          update_every=1, chunksize=10000, passes=1)
82     ldaOut=lda.print_topics(100)
83     lda.save("./models/ldamodel")
84
```

图 5-14 LDA 模型训练核心代码


```

46 sources = {
47     'clean-corpus/test-neg.txt': 'TEST_NEG',
48     'clean-corpus/test-pos.txt': 'TEST_POS',
49     'clean-corpus/train-neg.txt': 'TRAIN_NEG',
50     'clean-corpus/train-pos.txt': 'TRAIN_POS',
51     'clean-corpus/train-unsup.txt': 'TRAIN_UNSUP'
52 }
53
54 sentences = LabeledLineSentence(sources)
55
56 model = Doc2Vec(min_count=1, window=10, size=100, sample=1e-4, negative=5, workers=8)
57
58 model.build_vocab(sentences.to_array())
59
60 for epoch in range(10):
61     model.train(sentences.sentences_perm())
62
63 model.save('./imdb.d2v')
64

```

图 5-15 Doc2vec 模型训练核心代码

```

6 sc = SparkContext()
7 corpus = sc.textFile('douban').map(lambda s: s.split())
8
9 def gradient(model, sentences): # executes on workers
10     syn0, syn1 = model.syn0.copy(), model.syn1.copy()
11     model.train(sentences)
12     return {'syn0': model.syn0 - syn0, 'syn1': model.syn1 - syn1}
13
14 def descent(model, update): # executes on master
15     model.syn0 += update['syn0']
16     model.syn1 += update['syn1']
17
18 with DeepDist(Word2Vec(corpus.collect())) as dd:
19     dd.train(corpus, gradient, descent)
20

```

图 5-16 梯度下降分布式计算核心代码

5.4 系统测试与验证

系统测试分为准确度测试和性能测试。准确度测试是针对文本分析模块中文本倾向分析算法的测试，采用测试方式是通过训练语料和测试语料的对比并绘制 ROC 曲线来测试算法的准确度。性能测试主要是针对 Spark 平台并行化效率的测试，该测试主要是针对大数据场景进行的效率测试，采用脚本监控的方式对比在单机环境和在集群环境下对相同语料处理的效率来产生测试结果。

5.4.1 准确度测试

ROC (Receiver Operating Characteristic) 曲线是用来评价一个二值分类器(binary classifier)的优劣，与其对应的是 AUC(Area Under Curve)值，在 ROC 曲线上 AUC 值被定义为 ROC 曲线下的面积，这个面积的数值不会超过 1，其用来标识分类模型的准确度，AUC 值可以直观的评价一个二值分类器的优劣，AUC 值越大表示分类器越好，反

之，分类器越差。

ROC 曲线的横坐标为 FPR(false positive rate)值，纵坐标为 TPR(true positive rate)值。其中，TPR 和 FPR 的定义如下：

$$\text{TPR} = \frac{TP}{TP + FN} \quad (\text{表示分类器预测的正类中实际正类的比例})$$

$$\text{FPR} = \frac{FP}{FP + TN} \quad (\text{表示分类器预测的负类中实际负类的比例})$$

其中，有如下定义：

- TP (True Positive)：实际为正类且被预测为正类的样本数，
- FN(False Negative)：实际为正类，但被预测为负类的样本数，
- FP(False Positive)：实际为负类，但被预测为正类的样本数，
- TN(True Negative)：实际是负类且被预测成为负类的样本数。

根据这四种情况，可以得到如下的表格：

表 5-3 ROC 混淆矩阵 (Confusion matrix)

		真实值		总数
		p	n	
预测值	p'	TP	FP	P'
	n'	FN	TN	N'
总数		P	N	

由以上混淆矩阵不难看出，精确度为 $\text{precision} = \frac{TP}{TP + FP}$ ，召回率为 $\text{recall} = \frac{TP}{P}$ 。根据 ROC

曲线我们可以得到 AUC 值，由于分类器的 ROC 曲线无法精确的给出分类器的优劣，所以可以通过 AUC 值来衡量一个分类的好坏，AUC 值具有以下意义：

AUC=1,说明该分类模型为完美分类器，采用该分类预测模型，存在至少一个阈值可以达到完美的分类，但是这是理想情况，大部分分类器模型的 AUC 值都小于 1。

$0.5 < \text{AUC} < 1$,说明该分类器的判断优于随机判断，如果设置合理的阈值，分类器就具有预测价值。

AUC=0.5,等于随机判断的概率，该分类器没有预测价值。

AUC<0.5,说明分类器比随机判断的概率还要低，但是，这不代表该分类器没有价值，如果 AUC 的值很低，可以通过预测与分类器结果相反的结果，可以达到优于随机判断的效果。

根据以上定义，本系统基于获取到的豆瓣电影评论语料集进行了模型准确度实验，用于训练的语料集被分为训练集和测试集两部分，其中训练集包括 12500 条正向电影评论文本，12500 条负向电影评论文本，50000 条未标注电影评论文本；测试集包括 12500 条正向电影评论文本，12500 条负向电影评论文本。同时对训练集训练基于 LDA 和

Doc2vec 的文本表示向量，并用 `SGDClassifier` 作为分类器，可以绘制分类模型的 ROC 曲线如下所示：

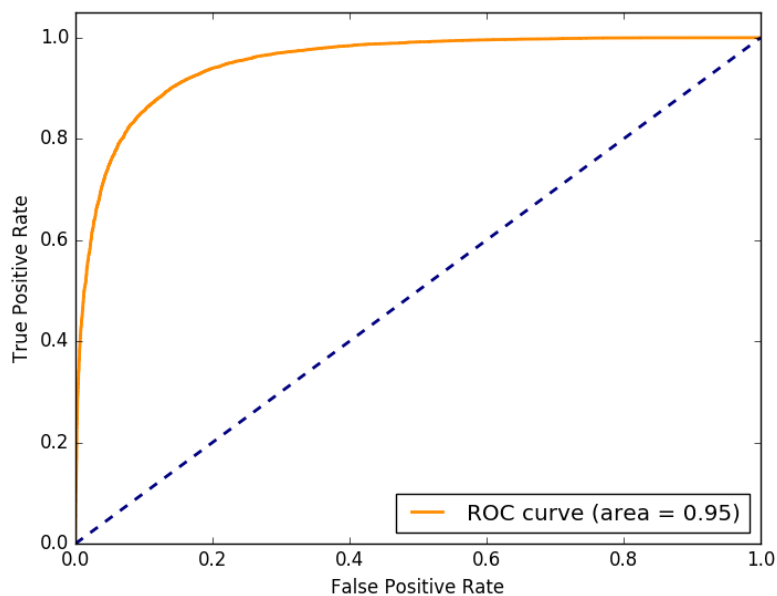


图 5-17 基于 LDA 和 Doc2vec 模型的 ROC 曲线图

由上图得到的 AUC 值为 0.95，同时，在同样的语料集上分别训练了 LDA 模型和 Doc2Vec 模型，在这两个模型上同样用 `SGDClassifier` 作为分类器可以得到的如下图的 ROC 曲线图：

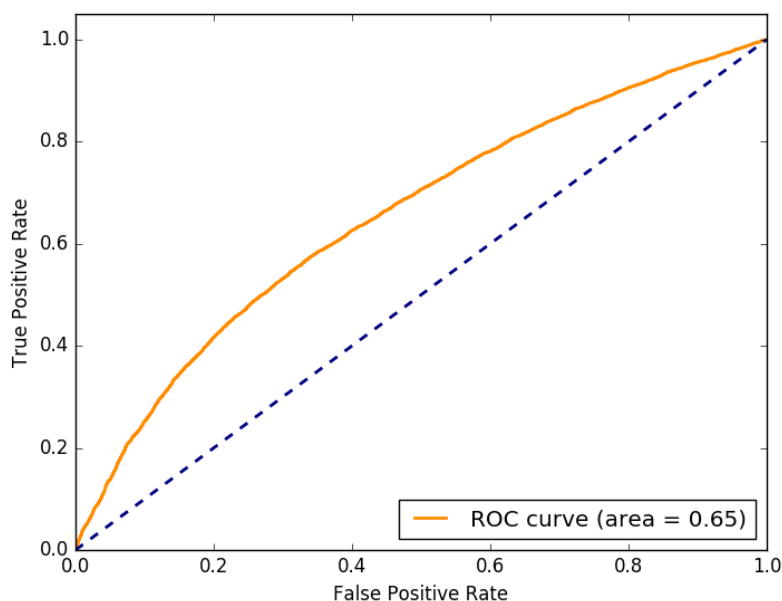


图 5-18 基于 LDA 模型的 ROC 曲线图

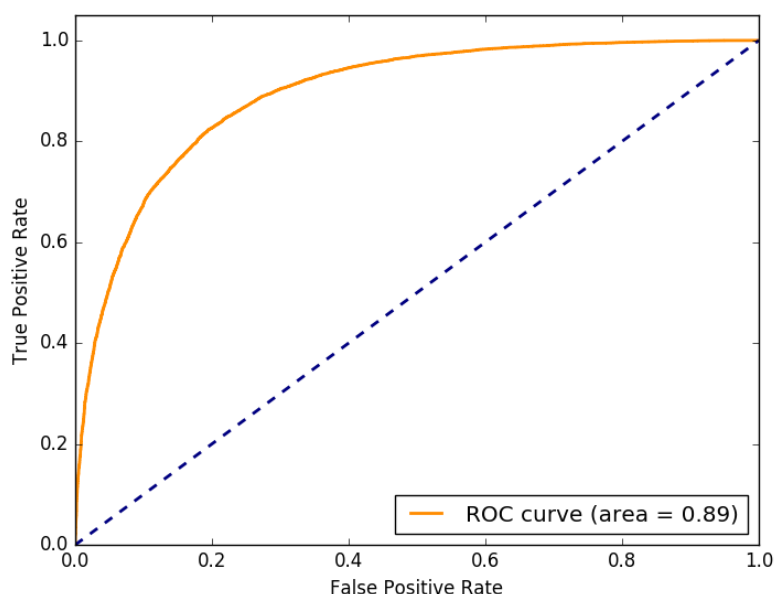


图 5-19 基于 Doc2Vec 模型的 ROC 曲线图

根据以上三个 ROC 曲线图可以看到，基于 LDA 和 Doc2vec 模型的文本特征表示方法在使用 SGD 作为分类器时可以得到该分类模型的 AUC 值为 0.95，而单单使用 LDA 模型或 Doc2vec 模型得到的 AUC 值分别是 0.78 和 0.89，该实验结果表明基于融合 LDA 模型和 Doc2vec 模型的文本特征表示方法可以更加准确的表示文本的特征，从而在处理文本分类问题时，提供更加准确的辨识度，而单单使用 LDA 模型或 Doc2vec 模型所得到的特征向量只是表示了文本的一部分特征，在文本分类过程中所得到的文本分类准确度有所下降，同时，Doc2vec 模型的 AUC 值高于 LDA 模型的 AUC 值，也表明 Doc2vec 模型对文本特征的表现力更强，这也是本系统采用 Doc2vec 模型作为主要文本表示模型，同时以 LDA 作为辅助模型映射到 doc2vec 向量空间的原因。

同时，与其他文本特征提取方法进行横向对比的结果如下表所示，从表中数据可以看到，采用传统的 TF-IDF 算法可以得到比较好的准确率，但是效果仍有很大的提升空间，而采用 PCA 进行特征提取的效果最差，其原因是 PCA 方法对于短文本而言很难统计出文本的特征，同时对比 Word2vec 和 Doc2vec 算法，其正确率没有较大的差距，这也说明了 Doc2vec 算法其实质是采用词向量作为基础，是对词向量模型的一种延伸，最后，我们采用结合 Doc2vec 算法和 LDA 算法的方式获得比较理想的准确率，这是因为充分考虑了文本中词与词之间的特征和文本与文本之间的特征，其训练的模型对于文本类型器有很高的辨识度，所以得到的准确率相较于其他算法效果很好。

表 5-4 与其他文本特征提取方法准确度对比表

模型	正确率
TF-IDF	86.80%

PCA	77.42%
LDA	83.55%
Word2vec	87.74%
Doc2Vec	88.33%
Doc2Vec+LDA	92.58%

5.4.2 性能测试

本节是针对并行化后的文本分析相关算法的性能测试，在本系统中总共对三个文本分析算法进行并行化处理，1.是对文本分词算法的并行化，2.是对 LDA 模型训练的并行化，3.是对 Doc2vec 模型训练的并行化。同时，通过与非并行化算法进行比较，可以看到经过并行化处理后，算法的训练速度得到提升，大大提高了文本分析的效率，缩短整个系统的文本分析时间成本。

为了得到比较准确的测试结果，本次实验采用多种语料集进行算法的训练，用于实验的语料集包含三部分，1.是通过爬虫爬取的豆瓣 top500 “热门电影”评论，其包含 500 部电影共 100,000 条评论文本，2.是搜狗语料库提供的网络评论语料库，3.是 imdb 电影评论语料库。同时，所有的并行化算法运行在由 4 个结点构成的 Spark 集群上，集群的服务器配置信息如下表所示：

表 5-5 Spark 集群实验环境配置表

服务器名	内存	CPU	线程
MASTER	8G	4 核	8 线程
NODE1-3	3x8G	3x4 核	3x8 线程

本实验采用加速比作为衡量并行化性能评价指标，加速比(speedup)指同一个任务在单处理器系统和并行处理器系统中运行消耗的时间比率，用来衡量并行算法或程序的性能指标，其定义如下：

$$S_p = \frac{T_1}{T_p}$$

其中， S_p 为加速比， T_1 是单处理器(单节点)环境下算法或程序的运行时间， T_p 是在有 p 个处理器(p 个运行结点)的环境下运行的时间。通过定义可以知道，加速比越大，说明算法的并行化效果越好。下图是通过运行在 Spark 集群上的分词程序的加速比测试结果图：

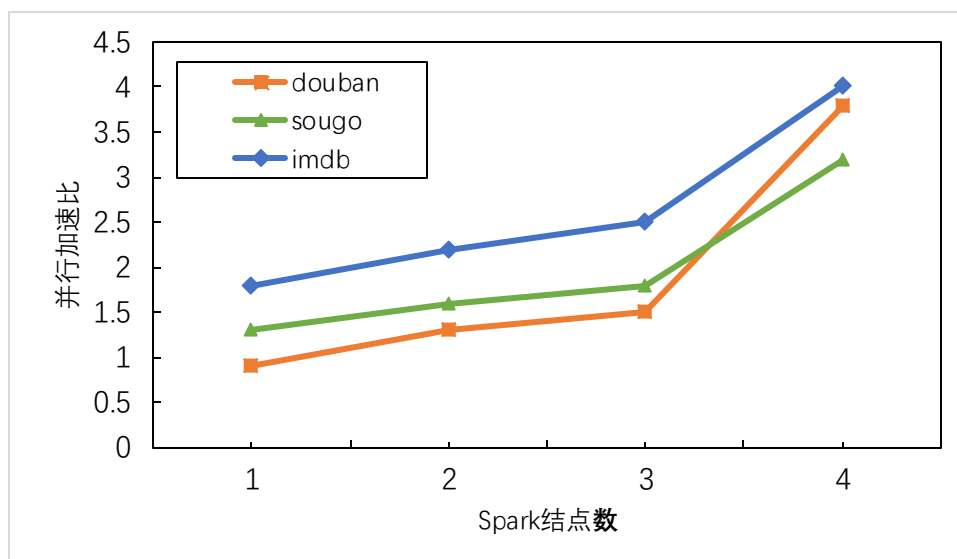


图 5-20 分词算法在 Spark 集群上的运行速率相较于单机环境有很大的提升

基于 Spark 集群并行化后的分词算法相较于单机环境下的分词速度有明显的提升，其原因是分词操作是天然的 Map 过程，其可以将问题拆解进行并行化运算，并充分应用各结点的计算能力，最后通过合并各计算结果而得到最终结果。同时，通过对 LDA 算法和 Doc2vec 的并行化可以提高整个文本特征训练过程的效率，下图是通过和单机环境做对比下两个模型的加速比曲线：

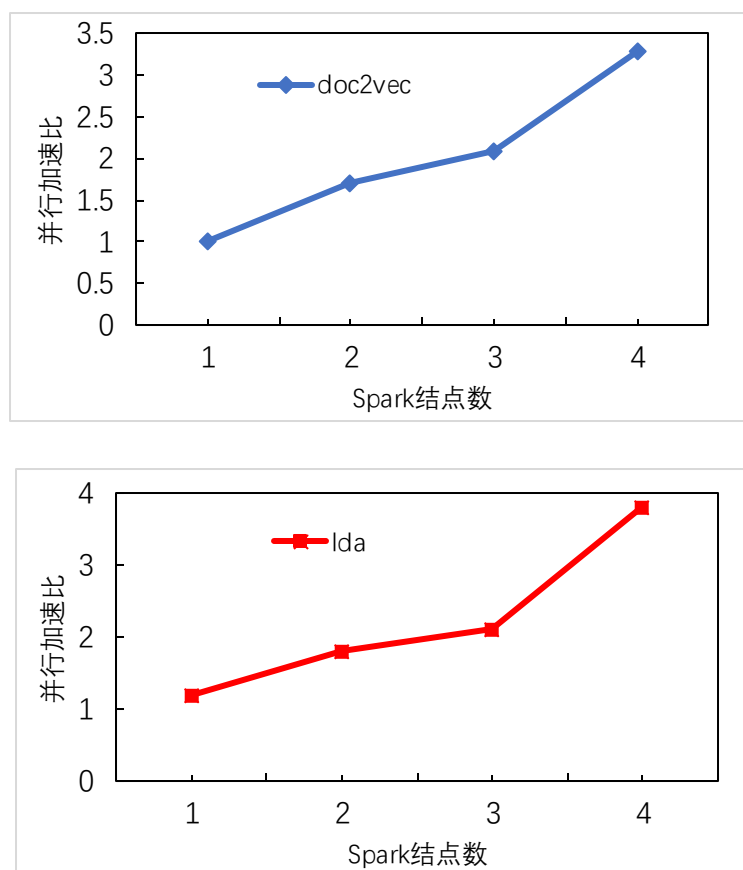


图 5-21 并行化的 lda 算法和 doc2vec 算法和单机运行情况加速比曲线

由上图可以看出，经过并行化处理后的算法随着结点个数的增加，算法的并行加速比也随之上升，因为在算法运行过程中，通过将可以并行化的计算结果分布到其他集群上，使得计算过程形成流水化计算，其计算速度受并行化后最慢的那个计算节点影响，但是，总体的计算时间相对于线性计算而言有较大提升。当集群个数达到 4 个结点时，并行加速曲线的斜率变大，说明当前的流水线速度与算法的计算过程相匹配，随着结点个数的增多，整个系统的加速比趋于平缓，当达到某一阈值后，由于结点之间通信和数据交换等原因，加速比会有下降趋势。

5.5 本章小结

本章是在第三章的基础上对整个系统各模块实现的阐述，以及对系统部署和测试的介绍。在第三章中，设计了系统的三个核心模块，分别是文本预处理模块、文本存储模块和文本分析模块。本章基于这三个模块对其进行详细的设计，主要包括模块程序流程图设计、核心功能模块展示，以及对代码进行分析。之后，本章对系统的整体部署进行了介绍，由于系统涉及的内容比较多，且依赖的模块比较复杂所以对系统的部署流程比较严格，本章给出了系统的部署流程图，并按照部署要求对系统进行部署。最后，本章

给出了对系统的验证测试结果，通过准确度测试和性能测试，分别展示了系统在文本分析算法的准确度上和系统运行性能上都有很大的提升。

第六章 结束语

6.1 论文总结

互联网的发展越来越迅速，随着科技的不断进步，人类将慢慢进入智能化时代，一系列基于文本分析的应用将逐渐影响着人们的生活方式。但是，目前文本分析领域仍然在探索之中，旧的研究方法已经不在适用于当前的需求，本文基于最新的研究成果，并结合大数据处理平台，对文本分析领域进行了系统的研究。

首先，通过研究最新的文本分析算法，并针对文本倾向分析这一应用场景，提出了采用 LDA 算法和 Doc2vec 算法相结合的文本倾向分析算法，该算法融合了文本的词语搭配特征和文本的统计特征，具有很高的辨识度。

其次，在 Spark 平台上将该算法并行化。本文设计了 LDA 算法的并行化模型，Doc2vec 算法的并行化模型，并将其运行在 Spark 集群上，利用 Spark 的并行化计算能力，提高了算法的计算效率，节约了系统的时间成本。

最后，以研究电影评论信息为例，设计了基于 Spark 平台的文本倾向分析系统，其中包括文本预处理模块、文本存储模块、文本分析模块，并通过实验证明该系统在准确率和效率上都有很大的提高。

6.2 下一步研究工作

在研究过程中，通过广泛阅读有关文本分析领域的最新成果和相关资料，以及通过积累丰富的实践经历，针对中文文本分析领域还有很多值得深入研究的地方。以下是关于中文文本分析领域几个重要的研究方面：

(1).针对文本语义分析的研究

本文是以文本倾向分析作为切入点研究的文本分析算法，除了文本倾向分析，还有文本语义分析，以及针对文本语义分析的应用场景，文本倾向分析实际上是文本的分类问题，基于对文本特征的提取与训练，然后基于模型的特征对文本进行分类，而基于语义的分析是更加高层次的文本分析任务，也是自然语言处理一直追求的目标，让机器真正理解人类语言，在语义识别过程中，还有很长的路要走，真正破解人类的语言密码还需要一段时间的深入研究，如果机器可以理解自然语言，那么，人工智能时代的到了也随之不远了。

(2).中文文本分析可视化研究

文本分析的可视化是指对文本分析任务中涉及的数据进行可视化展示,许多文本模型都可以用可视化的方式形象的展示出来,从而可以直观的提取文本的特征,对文本进行分类和聚类。文本分析可视化还是比较新的领域,在数据可视化中,占有重要的地位,如何直观高效的展示文本挖掘中得到的数据模型,语言特征,以及以什么样的方式来展示这些数据是文本分析可视化需要重点研究的问题。

参考文献

- [1] Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//ACM SIGOPS operating systems review. ACM, 2003, 37(5): 29-43.
- [2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.
- [3] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data[J]. ACM Transactions on Computer Systems (TOCS), 2008, 26(2): 4.
- [4] Wu L, Yuan L, You J. Survey of large-scale data management systems for big data applications[J]. Journal of computer science and technology, 2015, 30(1): 163-183.
- [5] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//2010 IEEE 26th symposium on mass storage systems and technologies (MSST). IEEE, 2010: 1-10.
- [6] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012: 2-2.
- [7] 王世泓,牛耕.基于情绪强度的中文微博情绪分析[J].计算机技术与发展, 2015,Vol.25 No.6: pages 138-140.
- [8] 陈永恒,左万利,林耀进.基于主题种子词的情感分析方法[J].计算机应用,2012,Vol.29 No.1: pages 48-52.
- [9] 杨亮,林原,林鸿飞.基于情感分布的微博热点事件发现[J].中文信息学报,2012,26(1):84-90.
- [10] 杨丽公,朱俭,汤世平.文本情感分析综述[J].计算机应用,2013,33(6):1574-1578.
- [11] 周立柱,贺宇凯,王建勇.情感分析研究综述[J].计算机应用,2008,28(11):2725-2728.
- [12] 周胜臣,瞿文婷,石英子,等.中文微博情感分析研究综述[J].计算机应用与软件,2013,30(3):161-164.
- [13] 陆文星,王燕飞.中文文本情感分析研究综述[J].计算机应用研究,2012,29(6):2014-2017.
- [14] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. journal of machine learning research, 2003, 3(Feb): 1137-1155.
- [15] Zheng X, Chen H, Xu T. Deep Learning for Chinese Word Segmentation and POS Tagging[C]//EMNLP. 2013: 647-657.
- [16] Le Q V, Mikolov T. Distributed Representations of Sentences and Documents[C]//ICML. 2014, 14: 1188-1196.
- [17] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of machine Learning research, 2003, 3(Jan): 993-1022.
- [18] 余永华,向小军,商琳.并行化情感分类算法的研究[J].计算机科学,2013,40(6):206-210.

- [19] 蒋婉婷. 基于Hadoop的中文微博主观情感分类的研究与发现[D]. 华东师范大学, 2014.
- [20] 卓可秋, 童国平, 虞为. 一种基于Spark的论文相似性快速检测方法[J]. 图书情报工作, 2015, 59(11): 134-142.
- [21] Dave K, Lawrence S, Pennock D M. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews[C]//Proceedings of the 12th international conference on World Wide Web. ACM, 2003: 519-528.
- [22] Das S, Chen M. Yahoo! for Amazon: Extracting market sentiment from stock message boards[C]//Proceedings of the Asia Pacific finance association annual conference (APFA). 2001, 35: 43.
- [23] Jin X, Li Y, Mah T, et al. Sensitive webpage classification for content advertising[C]//Proceedings of the 1st international workshop on Data mining and audience intelligence for advertising. ACM, 2007: 28-33.
- [24] Turney P D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews[C]//Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002: 417-424.
- [25] Pang B, Lee L, Vaithyanathan S. Thumbs up?: sentiment classification using machine learning techniques[C]//Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002: 79-86.
- [26] 赵军, 许洪波, 黄莹菁, 等. 中文倾向性分析评测技术报告[J]. 第1届中文倾向性分析评测研讨会论文集. 北京:[出版者不详], 2008: 1-20.
- [27] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. journal of machine learning research, 2003, 3(Feb): 1137-1155.
- [28] Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch[J]. Journal of Machine Learning Research, 2011, 12(Aug): 2493-2537.
- [29] Frome A, Corrado G S, Shlens J, et al. Devise: A deep visual-semantic embedding model[C]//Advances in neural information processing systems. 2013: 2121-2129.
- [30] 《计算机自然语言处理》王晓龙[M], 北京: 清华大学出版社, 2005-04-01.
- [31] Mikolov T. Statistical language models based on neural networks[J]. Presentation at Google, Mountain View, 2nd April, 2012.
- [32] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.
- [33] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.
- [34] 庞景安. Web 文本特征提取方法的研究与发展[J]. 信息系统, 2006, 29(3): 338-367.
- [35] 马兆才. 文本分类中的两阶段特征降维[J]. 甘肃科技, 2014, 30(20): 27-29.
- [36] 刘忠宝, 赵文娟. 融合全局和局部特征的文本特征提取方法研究[J]. 情报探索, 2016(1): 1-3.
- [37] July, 通俗理解LDA主题模型[EB/OL]. 2014-11-17. http://blog.csdn.net/v_july_v/article/d

- etails/41209515.
- [38] Rickjin, LDA 数学八卦[EB/OL]. 2013-02-08. <http://www.flickering.cn/tag/lda/>.
- [39] 《机器学习》周志华[M], 北京: 清华大学出版社, 2016-01-01.
- [40] Hofmann T. Probabilistic latent semantic indexing[C]//Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1999: 50-57.
- [41] Hofmann T. Probabilistic latent semantic analysis[C]//Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1999: 289-296.
- [42] Heinrich G. Parameter estimation for text analysis[J]. University of Leipzig, Tech. Rep, 2008.
- [43] Leisink M A R, Kappen H J. Computer generated higher order expansions[C]//Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 2002: 293-300.
- [44] Peghoty. word2vec 中的数学原理详解[EB/OL]. 2014-07-19. <http://blog.csdn.net/itplus/article/details/37969519>.
- [45] 《数学之美》吴军[M], 北京: 人民邮电出版社, 2012-05-01.
- [46] 王磊. 一种高性能 HDFS 存储平台的研究与实现[D]. 西安电子科技大学, 2013.
- [47] 柳平, 李春青, 姬婵娟. 基于 HDFS 的云存储架构模型分析[J]. 电脑知识与技术, 2013, 36: 092.
- [48] 邹振宇, 郑焱, 王嵩, 等. 基于 HDFS 的云存储系统小文件优化方案[J]. 计算机工程, 2016, 42(3): 34-40, 46.
- [49] Le Q V. Building high-level features using large scale unsupervised learning[C]//2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013: 8595-8598.
- [50] Dahl G E, Yu D, Deng L, et al. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2012, 20(1): 30-42.
- [51] 吴振华. 分布式词向量的研究及其并行化[D]. 中国科学院大学, 2015.
- [52] J Dean, GS Corrado, R Monga, K Chen, M Devin, QV Le, MZ Mao, M'A Ranzato, A Senior, P Tucker, K Yang, and AY Ng. Large Scale Distributed Deep Networks[C]. NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada, 2012.
- [53] Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. arXiv preprint arXiv:1603.04467, 2016.