密级：　　保密期限：

北京郵電大學

硕士学位论文

题目：基于社交网络情感分析的股价实时预测系统的设计与实现

学　　号：_____

姓　　名：_____

专　　业：计算机科学与技术

导　　师：_____

学　　院：网络技术研究院

2019 年 01 月 07 日

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Thesis for Master Degree

Topic: Design and Implementation of Realtime Stock Prediction System Based on Social Network Sentiments Analysis

Student No.:
Candidate:
Subject: Computer Science and Technology
Supervisor:
Institute: Institute of Network Technology

January 7th, 2019

Student No.:

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解并同意北京邮电大学有关保留、使用学位论文的规定，即：北京邮电大学拥有以下关于学位论文的无偿使用权，具体包括：学校有权保留并向国家有关部门或机构送交学位论文，有权允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，有权允许采用影印、缩印或其它复制手段保存、汇编学位论文，将学位论文的全部或部分内容编入有关数据库进行检索。（保密的学位论文在解密后遵守此规定）

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

基于社交网络情感分析的股价实时预测系统的设计与实现

摘　要

股票市场变化作为资本市场波动的重要组成部分，投资者一直希望能掌握其变化规律。对股票市场走向的准确预测，不仅可以为投资者提供有效的投资建议，还可以为资本市场提供经济发展的指导意见，有利于资本市场的稳定与繁荣。但是，由于市场信息的完整披露以及交易的自由使得股价波动具有随机性，股价预测一直面临着许多困难。尽管如此，考虑到投资人的非理性情感因素，即对上市公司的情感倾向会影响上市公司的市场表现，进而导致其股票回报发生变化。因此，在短期内对股价进行一定程度的预测是可行的。为解决上述股价预测的问题，本论文设计并实现了一套基于社交网络情感分析的股价实时预测系统。

本论文改进了在线增减量支持向量机（Online Incremental Decremental SVM）算法的淘汰策略以提出了一种基于最短更新步长的在线被动攻击支持向量机（Online Passive Aggressive SVM）算法，并使用时间序列算法完成对股价的预测。预测系统由数据获取、数据传输、数据预处理、数据计算、数据存储以及数据展示等部分构成。本系统通过实时收集社交平台中上市公司舆论评价的情感信息，由高可用的传输系统发送给数据预处理系统进行预处理，然后通过计算平台完成计算分析。本系统最后将计算后的数据结合收集的历史股价数据完成实时预测，使用可视化工具展示预测结果。

本论文验证了流处理系统基础组件的可用性，提升了算法设计的准确率以及股价趋势预测的准确率等指标，并对整体系统进行了功能性以及非功能性测试。

关键词 流处理 情感分析 股价预测 在线学习

DESIGN AND IMPLEMENTATION OF REALTIME STOCK PREDICTION SYSTEM BASED ON SOCIAL NETWORK SENTIMENTS ANALYSIS

ABSTRACT

The variation of stock market, is one of the most important parts of capital market fluctuations, is always expected to be predicted by investors. The prediction of its variation provides not only useful advice for investors, but signals for economic stability and prosperity. However, since the complete market information is always exposed to the public and the trade is free, stock prices often show their randomness, and predictions are never easy to made. Nevertheless, not all the investors are rational enough, which makes that emotional tendency towards a specific company will definitely affect its market performance and return in stock prices. Therefore, to some degree, stock price prediction in a short period is not impossible. To address these prediction problems, the thesis designs and implements a realtime stock prediction system based on social network sentiments analysis.

The thesis proposes an improvement for the elimination strategy of an Online SVM algorithm, and proposes an Online Passive Aggressive SVM algorithm based on minimize single update step length. The thesis also implements time series algorithm on stock price prediction. This system is divided into several modules: data access module, data transport module, data pre-process module, data calculation module, data accumulation and storage module, data visualization module. By acquiring social network sentiment data towards specific listed companies, and transporting them via high availability network channel, the system can pre-process these data, and calculate them. Finaly, the system outputs and shows prediction results by accumulating sentiment data and history stock prices.

The thesis first validates the availability of infrastructures of the stream

system, then improves the accuracy of designed algorithm, finally validates the complete project.

KEY WORDS stream system sentiment analysis stock prediction online learning

目录

第一章 绪论

1.1 研究背景和意义

对股票市场的走向进行有效预测，一直以来是研究金融的专家学者所期望达到的目标。有效地预测股价，能为券商和投资银行等资产买方机构提供重要的参考价值，例如规避风险或者扩大资产收益，有利于资本市场的稳定与繁荣。为此，专家学者们提出了许多有关股票市场的金融学假说或者模型，他们希望这些假说和模型能够为股票的走势作出预判。目前，主流学术观点大致可以分为两个研究方向。

第一种是以有效市场假说[1]为基础理论的随机游走模型[2]。随机游走模型认为，股票市场的股价完全由市场上可以得到的信息反应出来。其前提是，股票能够因有效信息自由交易，且有效信息得到充分地披露[2]。由于所有交易人都拥有公开的信息（例如上市公司的财务报表等），且其交易完全自由，所以导致股票的市场行为变得完全随机[2]。在这一前提下，任何股票的涨跌情况都是不可知的。第二种假设与第一种假设相反，因为实际的股票市场并不完全和上述理想的假设一致。首先，股票市场有许多信息无法被完全披露，或者接收到信息的时间不完全同步。其次股票市场的交易人不能完全理性地交易[3]。例如 2008 年美国次贷危机发生前，美国房地产市场的交易者普遍认为未来商品房交易市场前景欣欣向荣，导致超量的资金投入到房地产市场。由于许多商品房购买人缺乏稳定的还贷能力，投入到房市的资金大部分转化为泡沫，导致波及全世界的金融风暴[4]。基于对上述两个假设的考虑，股票市场也许会有一些行之有效的预测方法。在 2013 年诺贝尔经济学奖得主罗伯特希勒的《非理性繁荣》一书[3]中，作者在上世纪末互联网行业泡沫的破灭前成功地预测了互联网股市的崩盘，作者也在 2007 年成功预测了美国房地产市场的崩盘。

在诸多的非理性因素中，除了对市场的错误预估等因素，上市公司的舆论对于股价的影响也十分明显[5,6]。在当今互联网快速发展的时代，任何信息或公众事件，其影响都有经过互联网被放大的可能性，并且其发酵速度之快与传播效率之高，总是远远超过大众的想象。而大量希望通过股票市场致富的股民由于出于对上市公司前景的考虑，进行股票交易时会很大程度上受到社交媒体情感倾向的影响，进而会对股票波动有可观的影响[7]。例如 2017 年 4 月 9 日美国联合航空爆出空乘殴打乘客丑闻[8]，大量用户在 Twitter 等社交媒体上以愤怒的情绪声讨美联航的暴行，美联航引发严重的公关危机。事件发生第二周，在纽约证交所上市的美联航股票暴跌 6%，导致 2.5 亿美元市值蒸发。

在主流金融学术界对股票走势的分析工作中，大部分都是基于上市公司的财务状况以及市场前景进行的综合分析工作，例如基于公司财报的市盈率、市净率、市销率等，或者基于行业轮动、资金走向等方向进行的分析[9]，分析周期往往较长。而目前学术界针对于短期的、基于社交网络情感的研究较少。在以往的研究中，Bollen 等人[7,10]利用 OpinionFinder 与 GPOMS 等工具，以 Twitter 的推文作

为数据来源，对推文进行情感分析，并对道琼斯指数（DJIA）进行了预测。但是Bollen 等人的研究有以下不足：

1. 研究的数据集来源于 2008 年 2 月至 12 月的推文历史数据以及道琼斯指数的历史数据，而非基于实时的股票市场与当前社交网络舆情的数据，其短期预测的指导意义十分有限；

2. 参与研究的反映股票市场变化的指标仅仅是道琼斯指数这一项指标。道琼斯指数由综合美国 30 家重大企业股价涨跌而得到，其低价股涨跌的权重远小于高价股涨跌的权重，且道琼斯指数一般用于评价美国整体工业的发展，无法对某些具体的上市公司进行评价；

3. Bollen 等人仅仅发现“Calm”这一情感因素对股票市场有显著关联，而 GPOMS 其他的 5 个情感并没有发现显著关系。

综上所述，Bollen 等人的研究在预测的实时性与预测对象的具体性上有一定的局限性。

在 Bollen 等人的基础上，Sul 等人[11]又对标普 500（S&P500)中具体的公司进行了类似的分析。Sul 等人定义了情感价（emotional valence）这一变量，利用线性回归的方法进行分析，并且按照 1 天短期与 10 天长期两种情况进行分类讨论，且得到了较好的结果。但是考虑到其线性回归的方法较为简单，可能存在着整体欠拟合的情况。而且，Sul 等人较好的结果也仅仅是某些参数下的某些指标较好，可能存在着局部过拟合的情况。另外，Sul 等人的工作也仅仅是对历史推特与股票数据进行的分析，而并不是基于实时舆情的预测分析，其对于短期股票价格波动预测的指导意义同样不大。

在该背景下，一个基于社交网络情感分析的实时股价预测系统更能符合当前研究的要求。首先，股票价格是投资者对上市公司未来回报预期的直接体现，作出对未来股价的预测，其现实意义比分析股票历史数据以及回溯股价历史走向要更为重要。其次，社交网络的情感与股票市场走向瞬息万变，在短时间内根据社交网络的情感倾向对特定上市公司的股价作出及时、合理的预测，符合在实际情况下的股票投资要求。

1.2 研究内容

为了解决上述研究背景下前人工作尚未解决的针对于特定公司的股价实时预测的问题，本文的研究目标为设计并实现一套基于社交网络情感分析的股价实时预测系统，能够根据实时的社交舆论情感倾向预测出特定上市公司的股价波动，为使用者提供参考。根据提出的研究目标，研究内容大致分为三个部分：

1. 研究实时社交网络数据的获取以及预处理方法；
2. 研究基于实时社交网络情感分析的股价预测算法；
3. 研究基于社交网络情感分析的实时流处理系统以及可视化系统。

三个重要的研究内容具体如下。

1.2.1 海量社交网络数据的获取以及预处理方法

开展研究首先需要解决数据获取的问题。数据获取可以依靠社交网站提供的

开放 API 完成。而社交网络平台每时每刻都会产生大量的文本数据，对其中的每一条文本数据都进行分析是不切合实际的，所以为了找到数据预处理的通用方法，需要找到能在数据分析前完成的数据间筛选的通用方法。

其次，社交网络媒体的文本中存在大量与传统文本的语法、词汇等有区别的表达（如原始文本中有许多利用 Unicode 编码格式表示的“emoji”表情符号）需要找到对应特殊的处理方法。

除此之外，通过数据获取工具获取到的社交网络文本数据往往是比较完整的格式化数据，但是并不是所有数据信息或者所有字段都是有用的，所以需要找到格式化数据过滤（即数据内筛选）的通用方法。

1.2.2 实时社交网络情感分析的股价预测算法

研究内容的主要工作为实时社交网络情感分析的股价预测算法设计、部署、训练以及评估。

一般而言，情感分析的通用方法即是传统的自然语言处理方法，包括建立情感词词库，学习分类文本，选择正确特征和探索上下文语境等几个方面[12]。

1. 建立情感词词库。建立情感词词库实际是选择合适的文本库并且选择合适的算法进行训练，将表述情感的词汇作为特征提取出来。在训练数据集的选择上，其文本数据需要符合网络用语的特征。例如，有情感标注的 Twitter 数据，亚马逊商品评论与评分数据等。

2. 文本分类的算法。研究文本分类算法主要是将自然语言数据转化为数学语言描述的数据，并且映射到情感词库中。

3. 选择正确的特征与探索上下文语境。选择正确的特征主要是对文本分类中的算法选择合适的特征以及参数进行训练。探索上下文语境主要是选择合适的模型表示上下文关系。

本文需提出一种股价预测算法，能简单有效地将情感分析结果反应到股价的波动上。该算法需能够结合相应上市公司财报预设，给定时间窗口内所有相关话题下社交网络舆论情感分析结果，以及股价数据的历史波动，完成对具体公司的较短时间尺度下股价涨跌的预测。而在股价预测算法评估时，预测结果可以直接与获取到的现实股价进行对比，并对预测的准确度、实效性进行分析，并找到一种能根据实际股价与预测值的差异对流处理数据粒度、算法模型、算法参数实时调整的方法。

选定了研究的算法或算法平台之后，需要考虑算法的训练以及部署的问题，这首先需要解决如何适应实时社交网络数据的问题。情感分析的训练数据类型可能相对多样化，但是都需来源于互联网的相关文本，以保证互联网语言的特征得以保留。

不同的情感分析算法需要使用合理的评估方案完成评估。由于情感分析属于二分类问题，应该考虑其准确率、AUC 以及 F1 Score 等基本的二分类指标。值得一提的是，其准确率等指标可不作为最主要的选择考虑指标。因为为解决更重要的实时数据的规模大、实时性要求的问题，评价指标应该主要考虑模型的适应性与运行时延。即算法部署后所耗费的计算时间（即响应时间）与算法系统可用性是需要特殊考虑的指标。

1.2.3 基于社交网络情感分析的实时流处理系统以及可视化系统

由于社交网络每时每刻产生大量的文本数据，经前期调研[13]的预计，其文本数据产生的速度可达 100MB/min。传统情感分析算法平台难以胜任该数量级数据的分析工作。有可靠性保证的高并发流处理平台才能满足本课题的需求。同时由于金融市场变化迅速，且传统的量化交易工具交易速度很快，预测需要实现较好的实时性。所以本文对于系统的计算速度有较高的要求。最后为了对海量数据的分析结果进行汇总，需要设计一套能够直观展示预测结果的平台。综上考虑应当采用流处理的系统作为基本的系统框架，大数据可视化平台作为展示平台。即本研究最后需要搭建一套基于社交网络情感分析的股价实时预测系统。

待实现的流处理实时预测系统应当关注整体的业务逻辑，系统的核心功能是面向业务的。即在有效的时间内完成流处理的预测任务是系统最重要的功能。（传统的机器学习平台仅关注整体的数据处理质量，是面向算法的。）除此之外，流处理系统还需要满足可靠性高的要求，传统机器学习平台如果发生宕机事故，重新启动运算即可。对于流处理系统而言，业务的无限可用是系统稳定性所追求的最高目标。实现该目标往往需流处理平台支持分布式部署能力以及支持容灾性设计保障。流处理平台往往还要解决高吞吐、高并发、高负载的需求，除了对硬件性能提出挑战之外，也对数据预处理方法的性能和核心业务逻辑提出了挑战。

1.3 论文的组织架构

论文主要由以下九章构成。

第一章，绪论。本章主要阐述本文的研究背景、研究目标、研究内容等；

第二章，相关理论以及技术。本章阐述相关的算法理论与技术，包括可能涉及到的情感分析股价预测算法以及系统可能用到的技术以及平台；

第三章，系统的需求分析。本章阐述相关系统的需求，包括功能需求和非功能需求。

第四章，基于流处理平台的股价预测系统的设计。本章阐述相关系统的设计，主要体现在架构设计和逻辑设计等；

第五章，社交网络在线情感分析以及股价预测的算法设计。本章阐述本文提出的新算法设计；

第六章，社交网络在线情感分析算法的评估。本章对第五章提出的算法进行评估以及与传统算法的比较；

第七章，基于流处理平台的股价预测系统实现。本章详细阐述系统实现的细节，以及实际应当考虑的技术实现；

第八章，基于流处理平台的股价预测系统的测试。本章阐述对于流处理系统的测试工作，包括功能和性能测试；

第九章，结束语。对工作完成部分作出总结，并且指出未来的研究方向。

论文的最后是参考文献以及致谢部分。

第二章 相关理论以及技术

本章主要介绍相关的情感分析与股价预测算法，以及利用到的流处理平台与可视化平台的技术。

2.1 相关算法理论

涉及到的算法理论包括词袋模型[14]、主成分分析（Principal Component Analysis, PCA）算法[15]、词频与反向词频（Term Frequency- Inverse Document Frequency, TF-IDF）算法[16]、word2vec 算法[17]、在线支持向量机（Online Support Vector Machine, Online SVM）算法[18]以及时间序列算法[19]等。

2.1.1 词袋模型

传统情感分析首先需要完成的是自然语言处理与文本分析工作。词袋模型是文本分析最基础的一种模型。词袋模型[14]将每个词作为一条索引，所有的索引组成一个集合，这个集合被形象地称为“词袋”。在这个集合中，统计每一条索引出现的频数即可得到词袋模型。利用简单的词袋模型，可以快速得到一段文本的词汇到一个高维向量空间的映射。其数学定义如下：

语料库中所有的词构成集合$W = \{w_1, w_2, \dots, w_{N_w}\}$，所有的句子构成集合$S = \{s_1, s_2, \dots, s_N\}$.

对于某个句子s_k,假设其词构成的集合为 $W_k = \{w_{k_1}, w_{k_2}, \dots w_{k_n}\}$，对应词汇出现的频率$F_k = \{f_{k_1}, f_{k_2}, \dots f_{k_n}\}$.

定义句子s_k的词袋向量为$BoW_k = \{W_k, F_k\}$.

2.1.2 2-Gram 模型

2-Gram 模型[20]是按照词汇在文本中的出现顺序，将前后出现的 2 个词汇组成复合词汇的模型。其数学定义如下：

语料库中所有的词构成集合$\mathrm{W} = \{w_1, w_2, \dots, w_{N_w}\}$，所有的句子构成集合$\mathrm{S} = \{s_1, s_2, \dots, s_N\}$.

对于句子集合中某个给定句子s_k,假设该二元词汇所构成的集合为$W_k^2 = \{\{w_{k_1}, w_{k_2}\}, \{w_{k_2}, w_{k_3}\}, \dots \{w_{k_{N-1}}, w_{k_N}\}, \}$，所对应二元词汇出现的频率$F_k^2 = \{f_{k_1}, f_{k_2}, \dots f_{k_{N-1}}\}$.

定义句子 s_k 的 2-Gram 向量为 $Bo^2W_k = \{W_k^2, F_k^2\}$

2.1.2 PCA 算法

PCA 算法[15]是一种文本特征提取方式。在高维的情况下，变量之间会存在多重共线性，即预测变量相互关联。多重共线性可能会导致解空间的不稳定与结果的不连贯，高维空间的稀疏也会导致变量之间的存在大量冗余的信息。降维可以将数据的主成分快速提取出来。通过对高维矩阵进行 SVD 分解可以得到一个矩阵的特征向量，该特征向量在高维空间所指示的方向可以使得原矩阵到该方向的方差最大，即该特征向量可以被看作矩阵的主成分。

词袋向量$BoW_k = \{W_k, F_k\}$（或 2-Gram 向量$Bo^2W_k = \{W_k^2, F_k^2\}$），通过酉矩阵分解求解出方程：

$$BoW_k \boldsymbol{\alpha}_j = \lambda_i \boldsymbol{\alpha}_i. \tag{2-1}$$

计算出二维投影：

$$p_j = v_j^T \phi(x_j) = \frac{1}{\sqrt{\lambda_i}} \boldsymbol{\alpha}_i^T BoW_{k\,j}. \tag{2-2}$$

2.1.3 TF-IDF 算法

TF-IDF 算法[16]是另外一种文本特征提取方式，用于评估某一特定的词汇对于一个文档集合的代表性。对于词汇$w_i \in W$或者$w_i \in W_k$，$i \in \{N_{s_1}, N_{s_2}, .., N_{s_l}\}$，即词汇$w_i$出现于某一批文章 $S = \{S_1, S_2, \dots, S_l\}$中，其词频为：

$$tf_i = \frac{n_i}{\sum_{j=1}^{l} n_{S_j}}. \tag{2-3}$$

其中n_i为w_i出现的总次数，n_{S_j}为 S 中第 j 篇文章的总词数，其反向词频为：

$$idf_i = \log\left(\frac{l}{1 + \{j : w_i \in W_{S_j}\}}\right). \tag{2-4}$$

其中，$\{j : w_i \in W_{S_j}\}$表示所有出现过w_i的文章数

二者相乘即得到 TF-IDF 的值：

$$tf - idf_i = tf_i * idf_i. \tag{2-5}$$

当 TF 越大的时候，说明词汇在某文档中出现的次数较多，当 IDF 越大时，说明该词汇仅在某文档中出现的程度越高。

2.1.4 Word2vec 算法

仅使用 PCA/TF-IDF 与 LR 回归作为分类算法的效果并不够。因为对词汇的统计仅仅考虑了词汇出现的频率，没有考虑词汇之间上下文关系，所以上述模型也就无法反映词汇本身的关联与含义。而由于在基于上下文的情感分析领域中 word2vec 算法的效果较好，于是考虑使用 Skip-Gram 与 CBOW 两种 word2vec 模型进行尝试[17]。两种模型的结构都是需要利用神经网络的结构。所以 word2vec 模型在训练时间上的消耗会明显增多，但预期的训练效果优于模型简单的 PCA/TF-IDF 算法。

CBOW 模型的训练输入是某一词汇上下文的词汇的词向量，输出是该词汇的词向量，即根据上下文的词汇推断该处的词汇。Skip-Gram 模型的训练输入是某一词汇的词向量，输出是给定词汇上下文的词向量，即根据该处词汇推断上下文的词汇。Word2vec 使用霍夫曼树代替传统神经网络中的隐藏层和输出层的神经元，其叶子结点起到输出层的作用，内部结点起到隐藏层的作用。输入层到隐藏层与传统神经网络采用线性变换加激活函数不同，而是对所有的输入词向量求和并求平均。隐藏层到输出层为了不计算所有词的 softmax 概率，采用霍夫曼树表示隐藏层与输出层。使用二元逻辑回归的方法，将霍夫曼树左子树定义为负类，右子树定义为正类，一般使用 sigmoid 函数进行判别：

$$P(+)=\sum_{+} x_w^T\theta = s(x_w^T)=\frac{1}{1+exp(-x_w^T)}. \tag{2-6}$$

在式（2-5）中，x_w^T是当前内部结点的词向量，θ是需要训练出来的模型参数。

2.1.4.1 Skip-Gram 模型

Skip-Gram 模型输入为文本库中某个词的词向量，训练对象为该词汇词上下文的词向量[17]。Skip-Gram 模型定义如下：

1. 定义语料库中所有的词构成集合 $W=\{w_1,w_2,\dots,w_{N_w}\}$，定义$W$的某一子集$Context(w)$构成$W$中的给定元素$w$上下文的词集合，集合大小为$m$;
2. 定义给定w对应的数据样本为$sample=\ \{Context(w),w\}$;
3. 定义输出层包含样本中词向量$v(w)\in R^n$;
4. 定义投影层为恒等投影$v(w)$.

然后以各词在语料中出现过的词当作叶子结点，以所有词 w 在语料中出现的频率作为权值构建霍夫曼树（Huffman Tree）T. T满足叶子结点N_w个，其他结点N_w-1个。

例如，在图 2-1 所示的霍夫曼树的结构中：

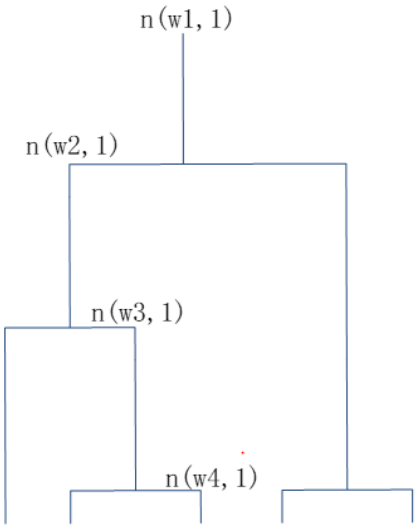

图 2-1 一个典型的 Huffman Tree 结构

以计算结点$w4$为例，计算的过程，实际上是期望最大化函数：

$$\prod_{i=3}^{3} P(\mathrm{n(w_i),i}) = \sum_{-} x_w^T\theta_1 \sum_{+} x_w^T\theta_2 \sum_{+} x_w^T\theta_3 . \tag{2-7}$$

通过最大化所有节点的似然函数乘积，便可以得到最后的迭代结果。具体做法是，每次仅用一个样本更新梯度，减少梯度计算量。得到w的对数似然函数是：

$$L = log\left(\prod_{j=2}^{l_w} P\left(d_j^w | x_w, \theta_{j-1}^w\right)\right)$$

$$= \sum_{j=2}^{l^w}\left(1-d_j^w\right) log\left[\prod_{j=2}^{l_w}\left(\left(1-d_j^w\right)\ log[\sum\left(x_w^T\theta_{j-1}^w\right)] + d_j^w\ \ log[\sum\left(x_w^T\theta_{j-1}^w\right)]\right)\right], \tag{2-8}$$

由此得到梯度表达式为：

$$\frac{\partial L}{\partial x_w} = \sum_{j=2}^{l_w}\left(1-d_j^w - \sum\left(x_w^T\theta_{j-1}^w\right)\right)\theta_{j-1}^w . \tag{2-9}$$

通过梯度上升法，进一步迭代求出θ_{j-1}^w和x_w。

2.1.4.2 CBOW 模型

CBOW 模型输入为文本库中某个词上下文的词向量，训练对象为该词汇词向量[21]。CBOW 模型定义如下：

定义语料库中所有的词构成集合$W = \{w_1, w_2, \dots, w_{N_w}\}$，定义$W$的某一子集$Context(w)$构成$W$中的给定元素$w$上下文的词集合，集合大小为$m$，定义给定$w$对应的数据样本为$sample = \{Context(w), w\}$.

定义在输出层包含$Context(w)$中的共$2m$个词向量分别为$v_1\left(Context(w)\right), v_2\left(Context(w)\right), \dots, v_{2m}\left(Context(w)\right) \in R^n$，$R^n$为词向量的向量空间。

定义投影层将$2m$个向量做求和累加，即

$$x_w = \sum_{i=1}^{2m} v_{2m} \in R^n. \tag{2-10}$$

以各词在语料中出现过的词当作叶子结点，以所有词w在语料中出现的频率作为权值构建 Huffman 树T，T中，叶子结点N_w个，其他结点N_{w-1}个。

2.1.4.3 Hierarchical Softmax

针对 Skip-Gram 与 CBOW 模型，采用 Hierarchical Softmax 的方法对问题进行简化[17]。在 Huffman 树T中，考虑某个出现过的词 w 对应的叶子结点：

1. 定义p^w为根节点出发到达w的路径；
2. 定义l^w为路径p^w的结点个数；
3. 定义$p_1^w, p_2^w, \dots, p_{l\hat{}w}^w$为路径$p^w$的所有结点；

4. 定义$d_1^w, d_2^w, \ldots, d_{1^w}^w \in \{0,1\}$为词$w$在$T$中的 Huffman 编码；
5. 定义$\theta_1^w, \theta_2^w, \ldots, \theta_{l^w-1}^w \in R^n$为$p^w$中非叶子结点对应的向量。

使用 sigmoid 函数作为激活函数，其词汇w条件概率的$p(w|Context(w))$ 为:

$$p(w|Contex(w)) = \prod_{j=2}^{l^w} p\left(d_j^w \theta_{j-1}^w\right). \tag{2-11}$$

其中：

$$p\left(d_j^w \theta_{j-1}^w\right) = s\left(x_w * \theta_{j-1}^w\right) \qquad if \;\; d_j^w = 0,$$

$$= 1 - s\left(x_w * \theta_{j-1}^w\right) \qquad if \;\; d_j^w = 1. \tag{2-12}$$

其对数似然函数为

$$L = \sum_{w \in C} log\left(\prod_{j=2}^{l^w} s\left(x_w * \theta_{j-1}^w\right)^{1-d_j^w} * \left(1 - s\left(x_w * \theta_{j-1}^w\right)^{1-d_j^w}\right)\right). \tag{2-13}$$

记

$$L(w,j) = \left(1 - d_j^w\right) log\left(s\left(x_w * \theta_{j-1}^w\right)^{1-d_j^w} * \left(1 - s\left(x_w * \theta_{j-1}^w\right)^{1-d_j^w}\right)\right). \tag{2-14}$$

采用随机梯度上升法，$L(w,j)$ 对于θ_{j-1}^w 的梯度为：

$$\frac{\partial L(w,j)}{\partial \theta_{j-1}^w} = \left[1 - d_j^w - s\left(x_w * \theta_{j-1}^w\right)\right] * x_w. \tag{2-15}$$

由此可以得到θ_{j-1}^w 迭代公式为：

$$\theta_{j-1}^w := \theta_{j-1}^w + \zeta\left[1 - d_j^w - s\left(x_w * \theta_{j-1}^w\right)\right] x_w. \tag{2-16}$$

其中，ζ表示学习率。

对于x_w的梯度计算为：

$$\frac{\partial L(w,j)}{\partial x_w} = \left[1 - d_j^w - s\left(x_w * \theta_{j-1}^w\right)\right] \theta_{j-1}^w. \tag{2-17}$$

x_w的迭代公式为：

$$v(w) := v(w) + \zeta \sum_{j=2}^{l^w} \frac{\partial L(w,j)}{\partial x_w} \qquad w \in Context(w). \tag{2-18}$$

进行迭代的伪代码为：

1. $e = 0$;
2. $x_w = \sum_{u \in Contex(w)} v(u)$
3. FOR j = 2: l^w DO{

   3.1 $q = s\left(x_w * \theta_{j-1}^w\right)$

   3.2 $g = \zeta\left(1 - d_j^w - q\right)$

3.3 $e := e + g\theta_{j-1}^{w}$

3.4 $\theta_{j-1}^{w} := \theta_{j-1}^{w} + gx_w$

}

4. FOR $u \in Context(w)$ DO{

$v(u) := v(u) + e$

}

在 Skip-Gram 模型中，推导相类似，θ_{j-1}^{u}与$v(w)$的迭代公式分别为：

$$\theta_{j-1}^{u} := \theta_{j}^{u} + \zeta\left[1 - d_j^u - s\left(v(w)^T\theta_{j-1}^{u}\right)\right]v(w); \tag{2-19}$$

$$v(w) := v(w) + \zeta \sum_{u \in Contex(w)} \sum_{j=2}^{l^u} \frac{\partial L(w, u\ , j)}{\partial v(w)}\ . \tag{2-20}$$

进行迭代的伪代码为：

1. $e = 0$;

2. FOR $u \in Context(w)$ DO{

3. FOR $j = 2: l^u$ DO{

3.1 $q = s(v(w)^T\theta_{j-1}^{w}))$

3.2 $g = \zeta\ (1 - d_j^w - q)$

3.3 $e := e + g\theta_{\{j-1\}}^{w}$

3.4 $\theta_{j-1}^{w} := \theta_{j-1}^{w} + g\ v(w)$

}

}

4. FOR $u \in Context(w)$ DO{

$v(w) := v(w) + e$

}

2.1.5 在线 SVM 算法

情感分析的过程是将一段文本的情感分为积极和消极两类进行求解，是一个典型的二分类问题。可以将词袋或者词向量以支持向量机的方法进行分类。传统的 SVM[22]是为了找出区分正负数据所在的超平面以及其支持向量。

定义社交媒体上舆论讨论的对象的集合为：

$$O = \{o_1, o_2, o_3, \ldots\ , o_m\}. \tag{2-21}$$

参与讨论的用户为：

$$U = \{u_1, u_2, u_3, \ldots\ u_n\}. \tag{2-22}$$

定义用户对于对象的评价为：

$$S = \{s_{i,j}\} \in\ R^{m*n}, \tag{2-23}$$

其中，

$$s_{i,j} = \phi(o_i, u_j). \tag{2-24}$$

$\phi(o_i, u_j)$为用户o_i对于u_j对象给出的评价，可能来源于 PCA 算法进行分类的结果或者 word2vec 词向量训练的结果。考虑对于对象u_j，当前时刻现有所有的对于该对象的语言评价到情感得分的矩阵记为：

$$w_j \in R^{m_1 * k} . \tag{2-25}$$

即对象u_j的情感得分为：

$$\boldsymbol{Y}_j = s_j^T * w_j. \tag{2-26}$$

定义该时间内新到来的的某一条/多条评价矩阵记为：

$$w_c \in R^{m_2 * k}. \tag{2-27}$$

不考虑新到来的评价，传统的基于软边界的 SVM 算法是找到能分割训练数据.$\boldsymbol{x}_j$的核的最佳函数的线性组合：

$$f(\boldsymbol{x}) = \sum_j \alpha_j w_j K(\boldsymbol{x}_j, \boldsymbol{x}) + b. \tag{2-28}$$

一般考虑最小化下面的目标函数：

$$\min_{0 \le \alpha_i \le C} : W = \frac{1}{2} \sum_{i,j} \alpha_i Q_{ij} \alpha_i - \sum_i \alpha_i + b \sum_i w_i \alpha_i , \qquad \forall i \in D. \tag{2-29}$$

一般使用拉格朗日对偶的 KT 条件求解，即满足：

$$\frac{\partial W}{\partial \alpha_i} = \sum_{\mathrm{j}} Q_{i,j} \alpha_j + w_j b - 1 = \ \mathrm{w}_j f(\boldsymbol{x}_i) - 1 = \begin{cases} \geq 0; & \alpha_i = 0 \\ = 0; & 0 < \alpha_i < C \\ \leq 0; & \alpha_i = C \end{cases}, \tag{2-30}$$

$$\frac{\partial W}{\partial b} = \sum_j w_j + \alpha_j = 0 . \tag{2-31}$$

考虑边界向量会随着新数据$i \notin D$加入模型而发生变化，为了保证 KT 条件能够持续成立，Cauwenberghs 等人通过改造其 KT 条件的目标函数，使得 SVM 模型能够跟随新加入的数据动态更新[18]。该算法对于在线系统与流式数据的支持性，这符合社交媒体流数据的要求。

首先记：

$$g = \frac{\partial W}{\partial \alpha_i}. \tag{2-32}$$

于是有：

$$\Delta g_i = Q_{ic} \Delta \alpha_c + \sum_{j \in S} Q_{ic} \Delta \alpha_c + w_i \Delta b, \qquad \forall i \in D \cup \{c\}, \tag{2-33}$$

$$0 = w_c \Delta \alpha_c + \sum_{j \in S} w_j + \Delta \alpha_j . \tag{2-34}$$

α_c是为了满足模型能够动态更新而加入的参数，初始值为 0，随着新加入模型的向量进行更新。由于需要保证$g_i = 0$，所以边界向量$S = \{s_1, s_2, \ldots s_{l_s}\}$的参数

α_s必须要满足：

$$\boldsymbol{Q}\begin{bmatrix}\Delta b\\ \Delta\alpha_{s_1}\\ \vdots\\ \Delta\alpha_{s_{l_s}}\end{bmatrix}=-\begin{bmatrix}w_c\\ Q_{s1c}\\ \vdots\\ Q_{s_{l_s}c}\end{bmatrix}\Delta\alpha_c, \tag{2-35}$$

其中Q为非半定矩阵对称矩阵：

$$\boldsymbol{Q}=\begin{bmatrix}0 & w_{s_1} & \cdots & w_{s_{l_s}}\\ w_{s_1} & Q_{s_1s_1} & \cdots & Q_{s_1s_{l_s}}\\ \vdots & \vdots & \ddots & \vdots\\ w_{s_{l_s}} & Q_{s_1s_{l_s}} & \cdots & Q_{s_{l_s}s_{l_s}}\end{bmatrix}. \tag{2-36}$$

可以记：

$$\begin{cases}\Delta b=\beta\Delta\alpha_c\\ \Delta\alpha_j=\beta_j\Delta\alpha_c\end{cases} \tag{2-37}$$

边界向量S需要满足条件消去 $\Delta\alpha_c$ 之后可以变形为：

$$\begin{bmatrix}\beta\\ \beta_{s_1}\\ \vdots\\ \beta_{s_{l_s}}\end{bmatrix}=-\boldsymbol{\mathcal{R}}\begin{bmatrix}w_c\\ Q_{s1c}\\ \vdots\\ Q_{s_{l_s}c}\end{bmatrix}. \tag{2-38}$$

其中$\boldsymbol{\mathcal{R}}=\boldsymbol{Q}^{-1}$

于是，Δg_i可以变形为：

$$\Delta g_i=\left(Q_{ic}+\sum_{j\in S}Q_{ij}\beta_j+w_i\beta\right)\Delta\alpha_c. \tag{2-39}$$

记：

$$\gamma_i=Q_{ic}+\sum_{j\in S}Q_{ij}\beta_j\ \ +w_i\beta. \tag{2-40}$$

由于在 KT 条件下需要满足：

$$\Delta g_i\equiv 0, \tag{2-41}$$

所以边界向量需要满足：

$$\gamma_i=0,\forall i\in S. \tag{2-42}$$

按照一定规则更新模型、淘汰数据，可以实现对模型的动态更新。通过调整其中的模型参数修改时机，可以调整模型的修改的频率，从而控制流数据对于模型的影响程度。上述内容在算法设计环节会详细介绍。

2.1.6 时间序列算法

股价是一种典型的随时间变化的数据，当认为投资者都是理性且不受情绪波动影响——满足有效市场假说时，股价的走势存在自相关性[23]，所以可以使用时间序列的方法并引入情感评价作为影响因素进行分析。以平稳序列为例，最常使用的模型为自回归移动平均（Auto Regressive Moving Average, ARMA）模型[19]。

$$X_t=c+\sum_{i=1}^{p}\varphi_iX_{t-i}+\varepsilon_t \tag{2-43}$$

其中X_t为当前时间窗口预测值，ε_t为引入的情感分析影响，c为股价的起始常数，p为选择的时间窗大小，φ_i为历史数据对当前股价影响程度。

2.2 相关技术

本节主要介绍 Netty 高可用网络通信框架技术[24]、Kafka 消息队列技术[25]、Spark Streaming 流处理平台技术[26]以及相应的存储技术和 Web 技术等。

2.2.1 Netty

Netty[24]是基于 Java NIO 的简单易用的网络通信框架，Netty 有并发量高，传输时延低，封装便捷等优点，非常适合本课题复杂网络情况下高并发、高可用的要求。Netty 的优势是：用户层的应用程序读取内核的缓存区的社交网络数据时，不需要进行多余的复制，而是直接在 ByteBuf 中进行读写。这确保了海量社交网络数据在发送时，能够最大化利用应用程序的资源。

在本课题中，主要通过自行定义 ChannelHandler 链式连接完成读写前、读写中与读写后的逻辑。例如，读取到的来自于 Twitter 的数据以 JSON 的形式封装，然后在发送前加上自行定义的 Netty 报文头，规定不同的关键词以区分 Twitter 数据的主题。在接收端拆包得到 Netty 报文头后，可以按照关键词发送到对应主题的消息队列中。

2.2.2 Kafka 消息队列

Kafka 消息队列[25]是一种高吞吐量的分布式消息订阅系统。社交网站上产生文本数据往往不能很快被计算，需要暂存且保证将来按照生成的顺序进行计算消费，Kafka 消息队列能很好地满足该要求。Kafka 支持亿级的消息堆积，并且可以通过重置偏移量，重新计算消费。

而且由于不同公司的关键字存在不同的主题与之对应，系统需要能按照不同主题进行订阅式消费。Kafka 支持订阅模式，对不同文本可以提前进行分类，这使得不再需要对文本进行主题聚类，减少了在情感分析结果映射到上市公司时所造成的误差。

除此之外 Kafka 比较轻量，且支持分布式系统，所以对本课题中实时预测系统的高可用、高并发以及快速开发支持较好。

2.2.3 Spark Streaming

Spark Streaming[26]是基于 Spark 的流处理平台。相对于传统的 Hadoop 平台，Spark 的 DAG 执行引擎支持在内存中对 RDD 这一数据结构进行迭代运算，基于 Spark 的 RDD，Spark Streaming 支持批处理的流运算。Spark Streaming 可以将社交网络的数据流按照时间间隔切分成 DStream 为单位的批数据，每一个 DStream 中都是社交网络文本中相应的词汇或者词向量等信息，通过提交多个 Job 组成的 Task，再按照引擎事先定义的 shuffle 逻辑，转化为新的 DStream 或者输出情感分析与股价预测的结果。

除此之外，Spark Streaming 还支持对长时任务的设计容错机制，如果出现社

交网络服务器不可用的连接超时错误，或者由于词向量计算长期无法收敛导致宕机等计算错误，只需重新计算一遍节点尚未提交的情感分析结果或者股价预测结果即可。Spark Streaming 的优势还在于，Spark Streaming 和 Spark 的集成效果好，Spark 上大量成熟的统计数学工具可以直接使用，以便尝试多种情感分析以及股价预测的方法。

2.2.4 Redis

Redis[27]是一种支持 Key-Value 的多种数据结构的存储系统。可用于缓存、事件发布或订阅、高速队列等场景。该数据库使用 ANSI C 语言编写，支持网络，提供字符串、哈希、列表、队列等数据结构，集合结构直接存取，基于内存，可持久化。

由于社交网络文本数据的规模庞大，每时每刻将计算中间量存储在计算内存中是不可能的，因为计算程序结束也意味着中间计算量全部丢失。所以系统可使用 Redis 存储情感分析的中间变量。Redis 支持良好的持久化方案以及对多语言的支持，易于部署。即使计算超时或者服务器宕机，情感分析与股价预测计算的中间量也能快速恢复。

2.2.5 Spring Boot

Spring Boot[28]是一个快速部署 Web 的框架，继承了 Spring Web 开发的完备性，同时不需要 Spring 复杂的配置，方便比较重量级的 Web 工程快速部署。Spring Web 是基于 MVC 的一个 Web 框架，通过配置 Spring 事务可以便捷地进行开发流程。对于股价预测结果的展示，可以采用 Spring Boot/MySQL 的方式，从 MySQL 中读取历史股价以及股价预测结果，以 Echarts 等框架使用 K 线图直观地展示出来。Spring Boot 可以省去 Spring Web 中 Spring MVC、数据库、事务、配置文件读取等工作，日志文件配置等工作，适合快速上手，且功能十分完备，适合未来其他与社交网络情感分析的实时预测系统拓展工作的部署。

第三章 系统的需求分析

为了详述地阐述系统设计以及算法设计的细节，本章对系统的需求进行分析。系统的需求分析分为功能需求和非功能需求两部分。功能需求分为数据层需求、业务层功能需求和系统层功能需求三部分。非功能需求主要为系统的性能和容量需求，阐述了系统所需要达到的性能指标和容量指标。

3.1 功能需求

功能需求按照“数据-业务-系统”的设计实现逻辑分为对应的三个部分。其中数据层需求定义了系统处理所需要的数据格式，业务层需求定义了面向用户需要完成的业务，系统层需求定义了实现业务需求与数据需求所需要的系统功能支持。

3.1.1 数据层需求

系统首先需要的数据为社交网络文本数据，使用 Twitter 开放 API 获取到的社交网络文本数据为 JSON 格式的数据。数据经过预处理系统，增加了关键字、关注度等信息，并删除了无用的信息，需要传输的数据格式如表 3-1 所示：

表 3-1 传输的推文数据格式

字段	内容	备注
id	1023407502342	推文 id
time_at	1542054455682	创建的 Unix 时间戳
text	“Good day for hiking, taking my new iphone. :)”	推文内容
hot	132	关注者数量+转推+点赞数量
keyword	“iphone”	关键词

数据经过预处理之后，系统需在 Netty 报文头添加表 3-1 中关键字信息，以字符串字节流的形式将数据通过通信框架发送给 Kafka 消息中间件，计算平台获取到表 3-2 格式的 JSON 字符串进行解析，同时存入 SQL 数据库与缓存数据库中。SQL 数据库的字段与表 3-2 的格式一致，缓存数据库直接以 JSON 字符串的形式储存数据，主键为推文 id。

数据在计算平台进一步计算，得到情感评分，并将关键字转化为对应上市公司，最后计算出数据淘汰时间，其格式需如表 3-2 所示：

表 3-2 中间计算的推文数据格式

字段	内容	备注
id	1023407502342	推文 id
time_at	1542054455682	创建的 Unix 时间戳
text	“Good day for hiking, taking my new iphone. :)”	推文内容
hot	132	关注者数量+转推+点赞数量
Company	“Apple”	关键词
expired	86400	数据淘汰日期
emotional rating1	0.62854	算法 1 情感评分
...		
emotional rating4	0.70254	算法 4 情感评分

最后将情感评分、提取到的股价历史数据经过进一步计算得到股价预测结果，并存入数据库，数据格式需求如表 3-3 所示。

表 3-3 股价预测结果的数据格式

字段	内容	备注
emotional rating	0.62582	情感评分
company	“Apple”	公司
window	8	时间窗口个数
prediction	150.3	预测变化

需要获取到的股价源数据字段如表 3-4 所示：

表 3-4 股价数据的数据格式

字段	内容	备注
open	125.2	开盘价
high	129.6	最高价
low	124.5	最低价
close	127.2	收盘价
volume	11,258,852	交易量
date	2018/10/12	日期

计算平台结合表 3-3 中前 3 项的储存内容，根据表 3-4 的股价数据可以计算出表 3-3 最后一项的股价预测数据结果。

3.1.2 业务层功能需求

系统的业务需求较为简单，即用户通过提交股价查询信息，能够获取多个公司最新的股价预测。从使用者角度出发，业务功能需求的内容虽然比较单一，但是在预测的形式上的需求仍可分为以下几个部分：

1. 可预测的范围广。从使用者角度而言，能预测的公司越多，预测越全面，本系统使用价值也就越大。

2. 预测的展示形式比较直观。预测数据应当以最简洁的方式，如图表的形式展示出来，需要数学计算进行归纳总结。

3. 预测的实时性较好。股票市场千变万化，舆论风向变化快，所以系统应当有对舆情做出快速反应的能力。

4. 预测的同时能查阅到历史数据。历史数据也是预测展示的一部分，从用户角度而言，结合历史数据能对实时股价预测结果来源有直观印象。

5. 预测的准确率需有最低标准。预测系统准确率是系统基本要求之一，考虑到股票的配额策略本身拥有容错特性，本系统的预测准确率应满足可实际应用的最低要求。

6. 预测服务一直可用。数据链路较长会提高服务不可用的概率，从使用者角度出发，服务应随时随地处于可用状态。

3.1.3 系统层功能需求

根据业务层功能的具体需求，预测系统需要满足获取、存储、处理、分析、展示等多个系统层的功能，具体如下：

1. 存储历史股价数据的功能。由于上市公司的股价表现与历史股价、上市公司的财报以及盈利表现息息相关，所以需要存储历史的股价数据，并在进行股价预测时，作为参数加入模型计算。

2. 实时获取当前股价数据的功能。当前股价的波动能很好反映上一预测周期结果的准确性。作为预测结果的真实值，当前股价需要在下一预测周期反馈给计算模型，用于在线算法的修正。同时当前股价需存入数据库，更新历史股价数据。

3. 实时获取指定关键词的数据处理的功能。每秒钟在推特网站上产生推特的数量大约在 10，000 条量级，同时获取该数量级的推特并同时进行分析是不现实的。通过追踪特定关键词，可以将所有讨论相关关键词的推特进行过滤收集，并按照关键词进行映射，可以得到对应上市公司的所有相关舆论评价的推特数据。

4. 数据的实时情感分析的功能。对指定上市公司舆论评价的推特数据进行情感分析并进行汇总，可以得到该公司在社交网络上的情感评价得分。

5. 分析给定公司的股价的功能。对于在纳斯达克与纽交所上市的指定公司，以及受舆论影响比较大的互联网公司，都应该被涵盖到该系统支持分析的列表中。对能预测的上市公司支持的越多，意味着其需要分析的数据越多，同时对每一家公司的分析结果越有说服力。

6. 数据可视化的功能。分析结果的展示效果应当是直观的。情感分析结果，应该有一个简洁直观的展示方式。上市公司的真实股价，应该用 K 线图以及其他简单的方式展示其预测走向。

3.2 非功能需求

除功能需求之外，系统还需要满足如性能需求和容量需求之类的非功能需求。系统的性能需求和容量需求本质都是系统对时延的需求[29]。

由于本系统需要处理的数据是海量的社交网络文本数据，首先需要面对的是波动的网络环境的问题以及高并发的业务挑战。具体而言，由于推特服务器的物理机房架设在美国，所以本地的推文数据获取系统可能会遇到波动性比较大的时延甚至超时。面对严重的时延时如何处理流量不均匀的流数据，以及面对超时时如何保证数据传输与计算的一致性，都是性能层面需要考虑的问题。

其次需要考虑的需求是，面对每秒查询速率（Query Per Second, QPS）可能达到 10^6 的实时数据，如何保证流数据能够持续地背压输出（即数据消耗能力满足数据生产的需求）而不产生消息堆积或者溢出，这是系统是否满足可用性的关键。

面对数据库频繁读写的问题，如何解决数据库 IO 的压力，如何解决数据库的并发事务带来的一致性的问题也是应当考虑的。如果使用分布式数据库，如何平衡一致性（Consistency）/可用性（Availability）/分区容错性（Partition Tolerance）三者之间的关系，也应当考虑。

除此之外，如何针对当前运行机器进行容量扩容，如何扩大追踪用户与公司的数量，应当作为系统容量的可拓展性的内容进行考虑。

最后，如果系统出现意外宕机导致局部甚至全局不可用，其容灾性保障系统如何快速从备份或者快照中恢复数据并选取策略进行重新计算，以及如何保证尚未保存的数据安全，这都是性能层面需要考虑的。

第四章 基于流处理平台的股价预测系统的设计

本章根据第三章中股价预测系统的需求分析进行具体的系统设计。系统设计分为架构设计、逻辑设计以及接口设计三个部分。其中架构设计主要按照系统层的业务层需求、数据层需求以及系统层需求分为了业务架构、信息架构和技术架构三个部分。逻辑设计主要阐述了基于上述三个架构具体的组件设计。接口设计阐述了不同组件之间通信接口的设计详情。

4.1 架构设计

根据软件设计规范，架构设计主要分为业务架构设计、信息架构设计以及技术架构设计三个方面。业务架构设计主要为业务层面的流程、规范设计等。信息架构设计主要为数据的种类、流向、功能设计等。技术架构设计主要为系统架构层面的结构功能设计等。

4.1.1 业务架构设计

预测系统的主要业务为股价的预测。根据上一节提出业务功能需求，业务架构设计如图 4-1 所示：

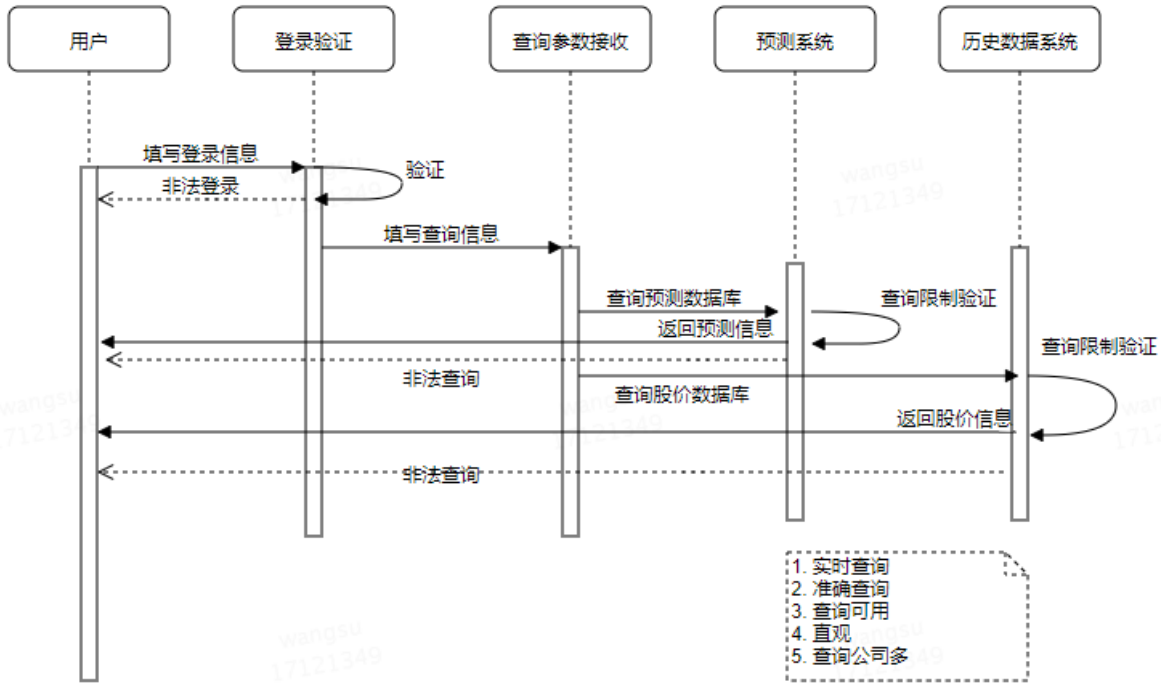

图 4-1 业务架构设计示意图

用户首先需要填写登录信息进行登录验证，非法登录的用户会被拒绝提供服务。登录状态的用户通过填写查询信息，将待查询的公司、时间等信息提交给系统。系统通过查询并返回预测结果和当前股价信息。

从业务逻辑上而言，用户查询需要处于登录状态，否则无状态的查询会为 Web 服务器带来流量风险。

除此之外，查询次数需要通过限制验证，否则外部接口查询也会为 Web 服务器带来流量风险。查询业务需要实现的 5 个内容与业务功能需求分析中提到的需求一致。

4.1.2 信息架构设计

用户端需要提供的信息主要为待查询的公司以及时间窗口，但是对于系统端，其数据链路比较长，需要处理以及存储的数据信息较多，其信息架构数据如图 4-2 所示。

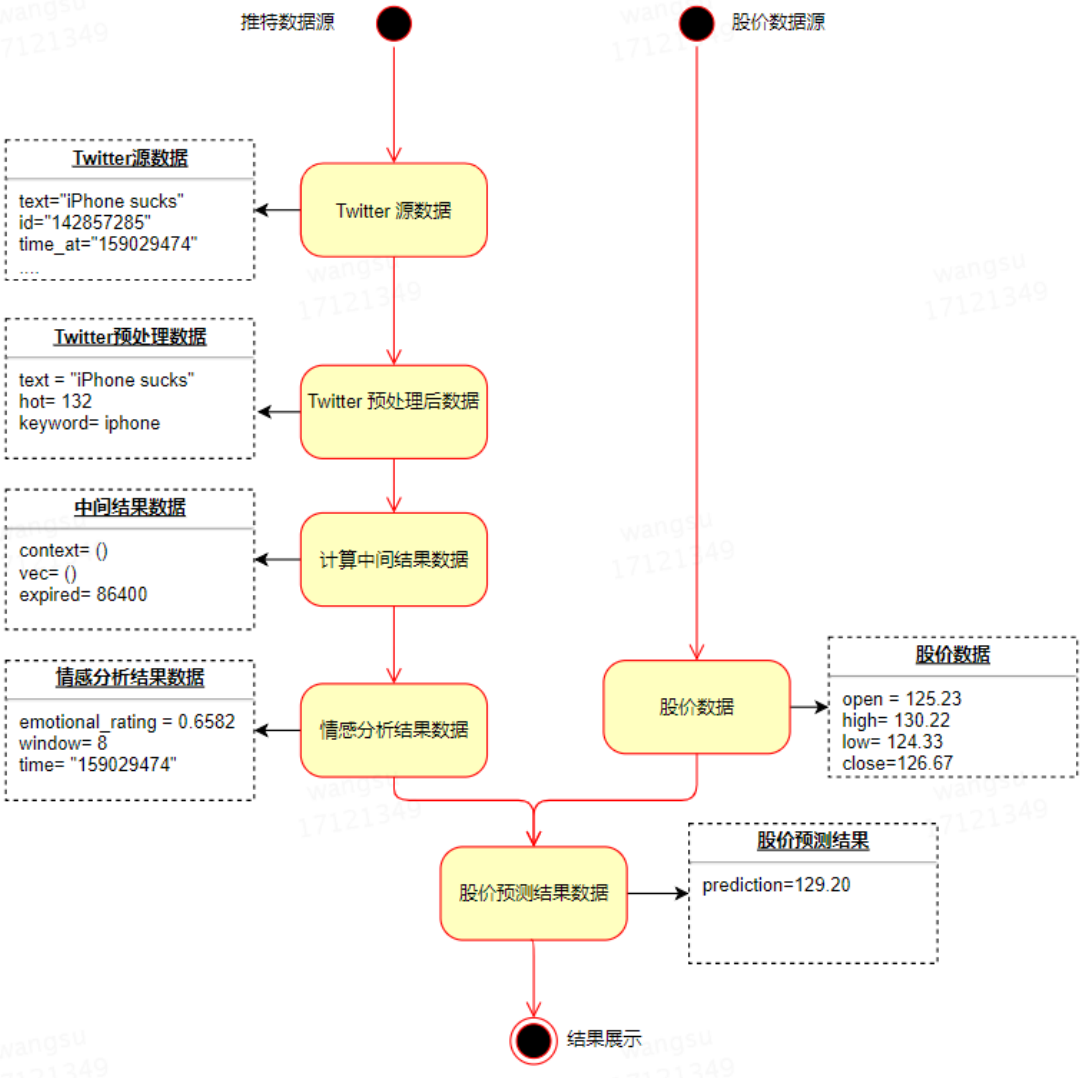

图 4-2 信息架构设计示意图

推特数据从获取到展示经过收集、处理、计算等处理，股价数据经过收集处理，然后系统汇总两项数据并计算，最后展示出预测结果。

4.1.3 技术架构设计

本系统技术架构上大致分为数据获取层、数据处理层、数据传输层，数据储

存层，数据计算层和数据展示层等六个层：数据获取层完成获取股价数据、推特数据获取，数据处理层完成数据的清洗、筛选（预处理），数据传输层完成数据的跨主机传输，数据计算层完成情感数据的计算以及股价预测计算，数据存储层完成数据的存储，数据展示层完成计算结果的展示。六个层次的结构大致如图 4-3 所示：

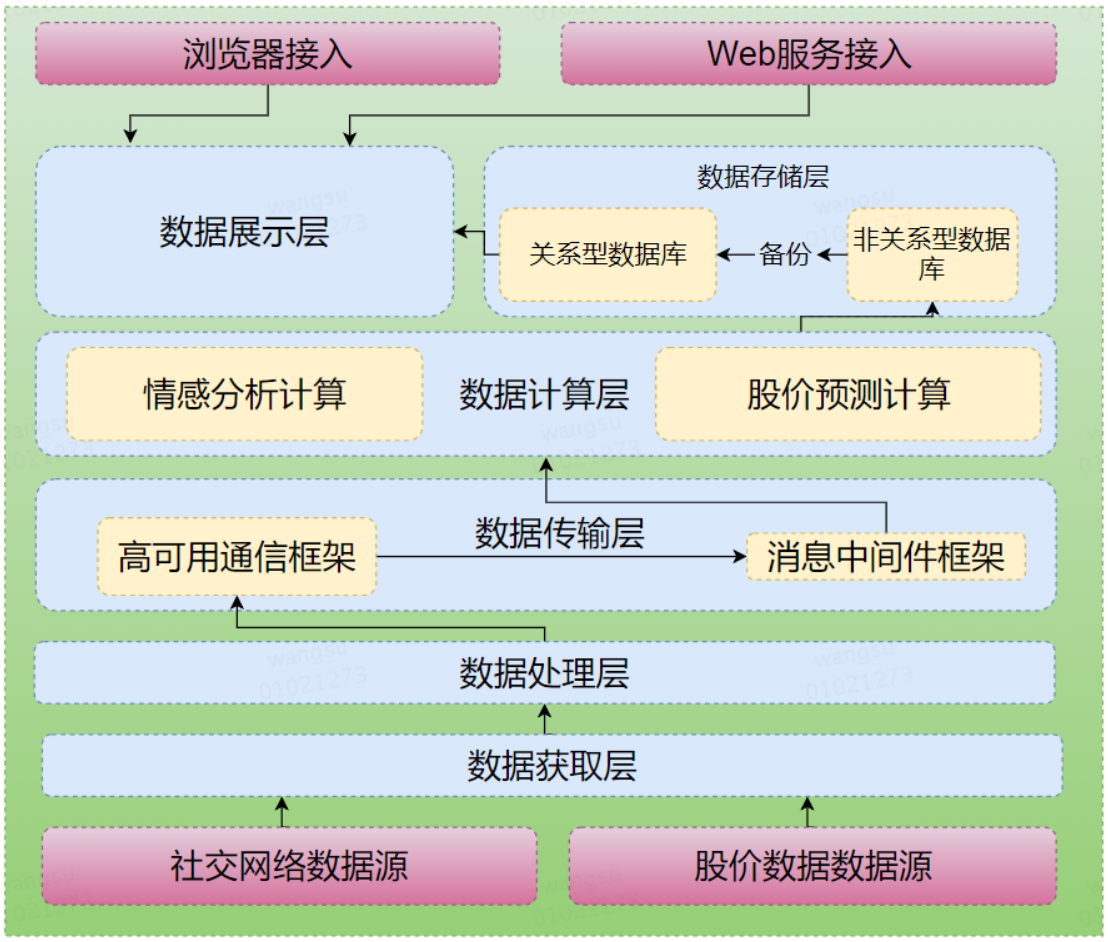

图 4-3 技术架构示意图

4.2 逻辑设计

4.2.1 系统设计

根据 4.1 中阐述的技术架构层的设计方案，结合 4.1 中信息架构的设计方案，技术架构中具体的模块选择以及数据流向设计如图 4-4 系统设计示意图所示：

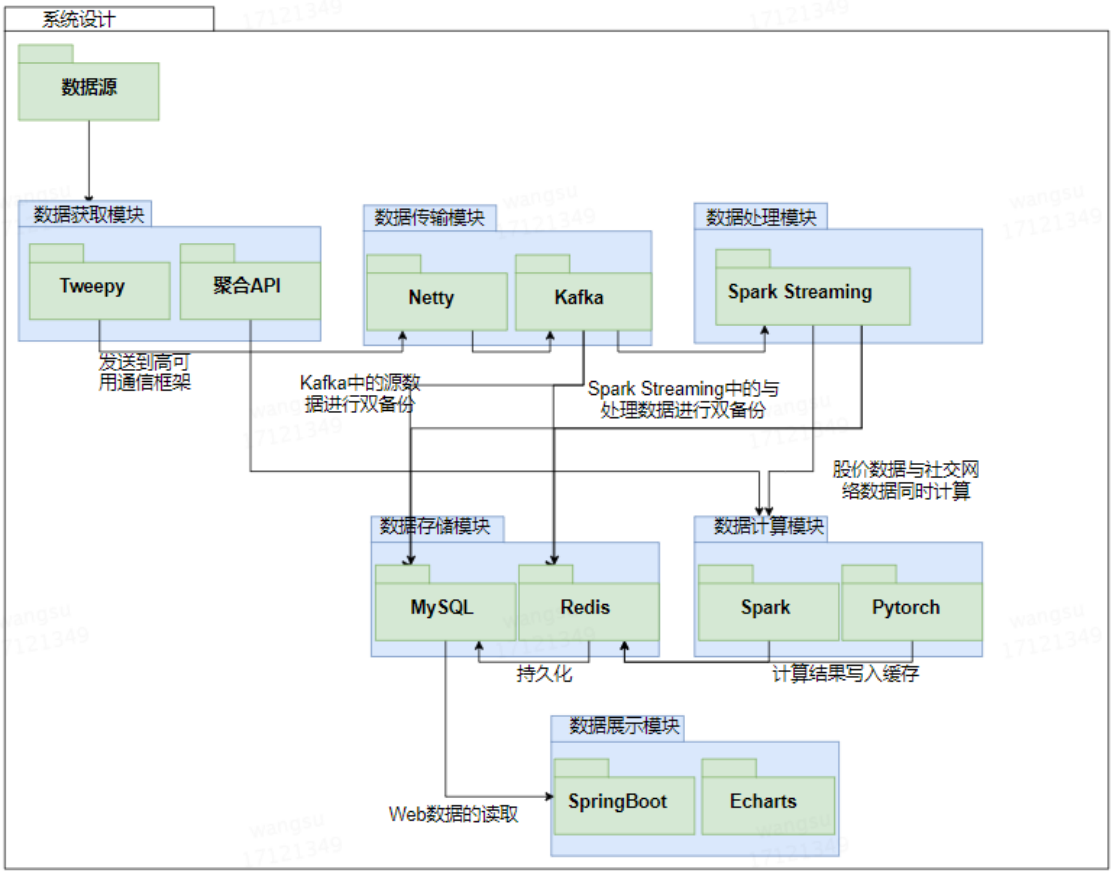

图 4-4 系统设计示意图

其中，数据获取模块包含 Tweepy 以及聚合 API 组件，将获取到的数据发送给高可用通信框架 Netty，Netty 添加报文头将消息发送给消息中间件 Kafka。Kafka 和 Netty 共同组成了数据传输模块。Kafka 中的消息由数据存储模块（MySQL, Redis）和数据处理模块（Spark Streaming）进行消费，数据存储模块同时也会存储来自数据处理模块以及数据计算模块（Spark）的中间变量和计算结果。最后数据存储模块将数据提交给数据展示模块（SpringBoot, Echarts）用于展示。技术架构中数据处理层的工作既包含了数据获取模块的数据筛选，也包含了数据处理模块的数据预处理。

4.2.2 组件设计

根据 4.2.1 中的系统设计，系统组件分为 Tweepy 组件与聚合数据组件、Netty 组件、Spark Streaming 组件、Kafka 组件、MySQL 组件、Redis 组件、机器学习（Spark）组件、SpringBoot 组件与 Echarts 组件等组件。

4.2.2.1 Tweepy 组件与聚合 API 组件

Tweepy 组件设计来源于推特官方提供的 API。根据推特的网站设计，推特对于不同的终端，例如移动端、Web 端等提供了统一的开放接口，用于用户的点赞，评论，发推等操作。开放接口统一由接口服务器进行管理。针对不同的账户操作，接口服务器在与后台逻辑交互的同时，也会为所有在 API 注册过的开发账号提供接口，将用户的操作的具体信息按照各个开发账号的具体要求发送给开发

者。上述内容即 Twitter 开放 API 能够提供的服务[15]。Tweepy 为第三方整合的 Twitter API 服务。其中，根据关键词获取对应推特就是 Tweepy 提供的功能之一，开发者可以向开放平台提供一定词汇组成的过滤器，平台会将包含对应词汇的所有推特以 POST 报文的方式转发给开发者。

聚合数据通过填写 HTTP GET 报文建立长连接获取股价数据。

4.2.2.3 Netty 组件

对于 Netty 组件的使用，主要包括传输序列化的工作、ChannelHandler 的改写等。原始数据以 JSON 的形式获取得到，精简后的数据依然保持其 JSON 的格式。考虑到对于网络时延波动的容忍，发送与接收端配置了负载均衡以及分布式系统进行数据接收。具体策略是，用于接收的分布式主机在发送端的 Zookeeper 注册接收服务，同时通过周期一定的心跳数据监控延迟。然后根据不同分布式主机的延迟分布进行负载均衡，延迟低的分布式机器在接收权重上更高。

4.2.2.4 Kafka 组件

Kafka 组件主要接收来自 Netty 的社交网络文本数据，数据源作为生产者加入消费队列。多台分布式主机同时注册 Kafka 消息生产者，其注册的主题按照获取推特时的关键字区分。关键字的获取来源于对 Netty 报文头的拆包得到。

4.2.2.5 Spark Streaming 组件

Spark Streaming 主要将从 Kafka 中订阅消费的推特进行一定程度的预处理。推特文本中会存在特殊符号（例如符号“:)”代表开心，例如“emoji”表情会以 Unicode 的方式展示），需要利用符号字典将这些特殊符号转化为对应的情感表达用语。网络用语还会有比较多的俚语，对于这些俚语往往在利用传统文本进行训练时，无法出现于正常的表达中，所以需要相应的俚语字典将俚语转换为正常的文本表达。除此之外，推文本身会存在附件或者链接的 URL，推文还包括了提醒的功能（用“@”符号标示，随后是一个用户 ID），转推推文甚至还会引用原推文等等。这些都是无效的文本内容，需要在预处理的时候进行相应的转换，或者直接进行丢弃。进行预处理后，Spark Streaming 还支持对于 PCA/TF-IDF 进行词汇统计，可直接生成 DStream 以表示的相应词袋向量，直接转化为 RDD 送入 Spark 机器学习平台或对应的 Hadoop 数据库中[16]。

4.2.2.6 MySQL 组件

系统使用 MySQL 作为关系型数据库，包含的分库分表有：

1. 股价数据库，包含了数十个上市公司的历史股价，字段包括股票代码，日期，最高价，最低价，开盘价，收盘价，以及总量。该库需要在可视化阶段提供展示历史的股价数据，以便更直观分析涨跌的效果。

2. 推文数据库，该库需要用于保存长时间段的全量推文数据（字段与先前数据需求分析中推文的 JSON 格式一一对应）以及关键性的源数据，例如对舆论影响较大的“大 V”的推文。推文数据库里保存的任何一条数据都会对在线算法产生影响，推文数据库里被舍弃的数据都不会再影响在线算法的模型。

3. 计算结果数据库，该库主要用于保存经过统计后的情感分析结果以及股

价预测结果。字段包括基于时间作为 Sequence 的 ID，上市公司的股票代码，涨跌的得分。本数据库主要用于 Web 平台展示。

4.2.2.7 Redis 组件

系统使用 Redis 作为非关系型数据库，需要建立包含的分库有：

1. 推文数据库，key 值为推文的 ID，value 值包括推文文本，推文时间戳，推文的作者信息。该库需要用于机器学习平台的灾后重新计算。基于 Redis 自身的热淘汰机制与 LRU 的规则，最不频繁使用的推文数据会被舍弃。当机器学习平台需要重新计算时，会直接遍历当前库里所有值，将数据重新加入计算节点。

2. 情感分析结果数据库，key 值为推文的 ID，value 是其情感得分以及爬取时的关键词。该库需要将机器学习平台的计算的结果首先写入 Redis，在 Redis 中缓存的情感结果通过一段时间的统计会写入 MySQL 数据库。如果出现计算平台宕机等不可用事故需要重新计算结果时，将原计算结果覆盖即可。同样根据热淘汰机制，最不频繁使用的计算结果会被舍弃。

4.2.2.8 机器学习组件

机器学习组件以 Spark 与 PyTorch 为主，Spark 平台的 MLlib 会按照事先训练的模型，对送入的词袋向量进行运算，同时基于 Python 的 PyTorch 工具会按照其他的模型（例如 word2vec 进行词向量的运算）进行运算，然后统一将中间量按照 LR 回归或者 SVM 进行二分类得到结果。最后将运算结果写入对应的数据库。

4.2.2.9 SpringBoot 组件与 Echarts 组件

Web 平台使用主流的 Spring MVC 作为 Web 框架，MyBatis 作为数据库的 ORM，采用 Echarts 作为可视化方案。

用户在 Web 页面选择待展示的公司以及预测的时间长度作为表单提交后，后台查询 MySQL 数据库，获取到的股价信息以及股价涨跌结果以 Echarts 提供的多种图表方式展示出来。

4.2.3 接口设计

模块之间的接口设计主要分为两类，HTTP 通信接口与 Socket 通信接口。

数据获取模块主要通过 HTTP 接口完成通信。数据请求方通过构造 GET 报文，HTTP 服务器与发送方建立长连接完成数据传输。数据展示组件也主要以 HTTP 通信，通过浏览器或者其他请求方的 GET 请求，将预测结果以可视化结果或者 JSON 报文的形式返回给请求方。

数据传输模块以及数据预处理模块主要以 Socket 通信接口完成通信。跨机房间的通信使用 socks5 协议作为 UDP 协议的下层协议，通过跳板服务器的转接，可以有效地应对由于机房的物理位置导致的网络延迟以及丢包的问题。

第五章 基于社交网络在线情感分析的股价预测算法设计

本章主要阐述了在线情感分析的股价预测算法的设计。流形式的社交网络文本数据，首先使用词袋等模型将自然语言转化为数学语言，然后采用在线 SVM[18] 等方法可实现对流数据的实时情感分析。本章提出了在线 SVM 算法适应于不确定数据集的淘汰策略，即通过拟合社交网络舆论倾向性的程度随时间的变化，动态地将情感因素作为影响因子加入到情感的预测模型中，或将之从模型中删除。接下来，通过对在线 SVM 算法的目标函数进行改造，提出了一种在线被动攻击 SVM 算法，该算法可以在一定条件下以最小化步长更新模型，同时最大化历史数据对模型的影响。最后，基于情感分析结果以及结果随时间变化的趋势，完成股价预测算法的设计。

5.1 在线情感分析的股价预测算法的背景

在第二章提到的 word2vec 算法的先前工作以及已经成熟的投入生产的工业化方案已经有比较好的表现。但是在分类器的选择上，先前工作中传统的 LR 分类器最大的问题在于：对于训练出来的词向量或者降维、变形的结果，其分类器模型仅仅与最开始选择的参数和训练数据有关[30]。所以 LR 分类器的参数以及训练数据选择十分关键。这是由于算法实际部署到流处理平台之后，新计算的数据不再能对分类器模型产生任何影响。除非新加入计算的数据能动态地对分类器模型进行更新，否则无法预估参数以及训练数据选择对后期算法的影响。

上述特殊情况在本研究中的具体表现之一是：尽管一家公司短期内所积累的历史舆论已经出现明显的负面倾向，股价出现持续走低的情况，但是由于公司的公关运营，公众对于所关注舆论点的淡忘以及注意力的转移，当前情感评价会体现为回升，股票会出现反弹。在数据计算层面，尽管历史数据会显示比较低的情感评分，但是由于新到来数据的急剧减少，会对于分类器模型的分类标准会产生比较明显的影响。即：历史数据的影响会随着热度淘汰明显降低影响力。

除此之外，部署在流处理平台的算法不再存在训练集的概念，其模型在下一次调整参数之前会相对固定，选择较稳定的模型进行部署会有一定难度。所以，本文提出了基于热度动态淘汰的 Online SVM 算法与在线被动攻击 SVM 算法。这两种算法能够较好解决新数据对模型的影响问题，新到来的数据能够及时加入模型中进行计算，且历史数据能够及时被淘汰，从而使得分类器模型能产生可控动态变化。

5.2 改进淘汰策略的 Online SVM 的设计

在第二章相关理论以及技术中提到的 Online SVM 算法将数据以流的形式加入模型，或者将数据从模型中删除，其加入与删除规则具体如下[18]。

每一次向模型中加入新到来的数据时，对于$\boldsymbol{Q}$与$\boldsymbol{\mathcal{R}}$的更新：

$$\boldsymbol{\beta} = \left[\beta, \beta_{s_1}, \dots, \beta_{s_{l_s}}, 1\right], \tag{5-1}$$

$$\boldsymbol{\mathcal{R}} \coloneqq \begin{bmatrix} & \mathcal{R} & & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \begin{matrix}0 & \cdots & 0\end{matrix} & & & 0 \end{bmatrix} + \frac{1}{\gamma_c}\boldsymbol{\beta}^T\boldsymbol{\beta}. \tag{5-2}$$

更新的步骤存在有反过程，即数据淘汰过程：

$$\mathcal{R}_{ij} \coloneqq \mathcal{R}_{ij} - \mathcal{R}_{kk}^{-1}\mathcal{R}_{ik}\mathcal{R}_{kj}. \tag{5-3}$$

综上总结，每一次向模型中加入新到来的数据时，计算其g_c以及α_c，并且对于重新计算的中间结果α_j与g_i需要考虑如下情况，并且按照如下流程更新：

1. 令$\alpha_c = 0$
2. $g_c > 0$，丢弃新到来的数据
3. 如果 $g_c \leq 0$，按照下面规则使得顺序使得增量α_c最大：
   - i. $g_c = 0$，将新来的数据c加入边界向量（支持向量），更新$\boldsymbol{\mathcal{R}}$，结束
   - ii. $\alpha_c = C$，将新来的数据加入错误向量集E，结束
   - iii. $0 \leq \alpha_j \leq C, \forall j \in S$, j 保持在S内，当左边等号成立j从S加入正常分类数据集R，当右边等号成立时，j从S加入E
   - iv. $g_i \leq 0, \forall i \in E$, i保持在E内，当等号成立时i从E加入S
   - v. $g_i \leq 0, \forall i \in E$, i保持在S内，当等号成立时i从S加入E

仅考虑更新的算法还无法适应本研究的要求，由于社交网络的舆论评价会有明显的涟漪效应[31]和淡忘效应[32]。涟漪效应是指舆论事件影响会随着传播不断扩大，针对该事件的情感会越来越强烈。即随着情感数据加入速度越来越快，模型受到新到来数据的影响会越明显。淡忘效应是指舆论事件随着讨论的冷却而迅速被公众遗忘。即数据淘汰随着时间逐渐明显。故该算法需要考虑新加入的模型的数据对原模型的影响程度，以及历史数据的淘汰速度。

首先针对于不同时期的数据以及不同影响力的作者的文章对于舆论的影响，引入自适应淘汰变量ζ，定义：

$$\zeta = \frac{1}{2}exp\left(-\frac{(C - |np(t)|)^2}{2}\right). \tag{5-4}$$

n为该文章的影响力，同样一篇文章，其在社交网络中的“话语权”越重要，认为该文章对于其评价对象的影响越明显。为了方便模型计算，影响力取为该账号的有效好友数与推文的点赞数转发数之和。

$p(t)$描述为一篇文章在社交媒体中的影响力随时间的变化。文章从发布到被普遍阅读其传播前半段近似指数分布，随着社交圈的扩散达到饱和，最后随着时间衰减逐步被淡忘。特点比较服从 Gamma 分布[33,34]，故考虑采用 Gamma 分布进行拟合：

$$p(t|\alpha, \beta) = \frac{\beta^{\alpha}t^{\alpha-1}e^{-\beta t}}{\Gamma(\alpha)}, \tag{5-5}$$

其中：

$$\Gamma(x) = \int_0^{+\infty} t^{x-1}e^{-t}dt. \tag{5-6}$$

不同参数的 Gamma 分布如图 5-1 所示：

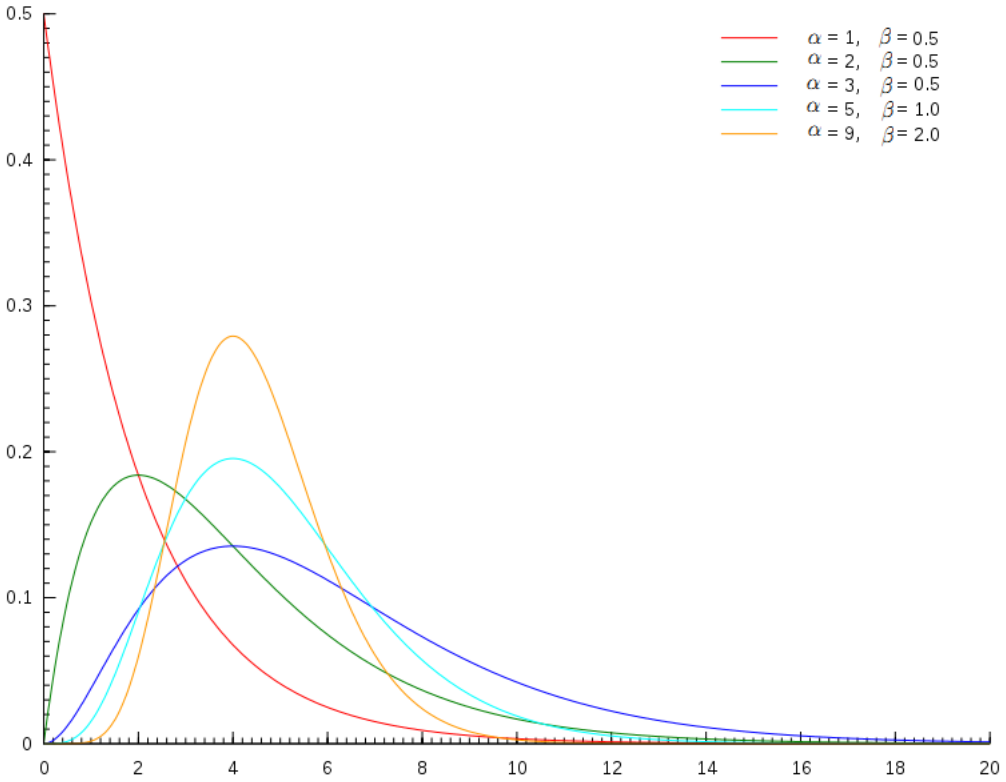

图 5-1 不同参数α β 下 Gamma 过程的图像

参数α决定了到达影响力最大值时的时间，参数β决定了到达的最大影响力。在拟合的训练阶段，定义任意t时刻的某个话题的影响力数据为：

$$\boldsymbol{p}_t = \{t, y_t\}, \tag{5-7}$$

其中：

$$\boldsymbol{y} = \{y_1, y_2, \dots, y_T\}. \tag{5-8}$$

$\mathbf{y}$为当前话题随时间变化的整体情感评价的真实值。拟合问题本身是一个非线性函数最小二乘拟合问题[35]。即：

$$\min_{\alpha,\beta} \sum_{t=1}^{T} \left(y_t - p(t|\alpha,\beta)\right)^2. \tag{5-9}$$

设$\boldsymbol{\gamma}^{(k)}$是解$\boldsymbol{\gamma} = \{\alpha, \beta\}$的$k$阶近似，考虑$p(\boldsymbol{\gamma})$在$\boldsymbol{\gamma}^{(k)}$附近的一阶 Taylor 展开式：

$$p^{(1)}(\gamma) = \nabla p\left(\boldsymbol{\gamma}^{(k)}\right)\gamma - \left[\nabla p\left(\boldsymbol{\gamma}^{(k)}\right)\boldsymbol{\gamma}^{(k)} - p\left(\boldsymbol{\gamma}^{(k)}\right)\right], \tag{5-10}$$

令

$$A_k = \begin{bmatrix} \nabla p\left(\boldsymbol{\gamma}^{(k)}\right) \\ \vdots \\ \nabla p\left(\boldsymbol{\gamma}^{(k)}\right) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial p\left(\boldsymbol{\gamma}^{(k)}\right)}{\partial \alpha} & \dfrac{\partial p\left(\boldsymbol{\gamma}^{(k)}\right)}{\partial \beta} \\ \vdots & \vdots \\ \dfrac{\partial p\left(\boldsymbol{\gamma}^{(k)}\right)}{\partial \alpha} & \dfrac{\partial p\left(\boldsymbol{\gamma}^{(k)}\right)}{\partial \beta} \end{bmatrix}, \tag{5-11}$$

$$\boldsymbol{b}=\begin{bmatrix}\nabla p\left(\boldsymbol{\gamma}^{(k)}\right)\boldsymbol{\gamma}^{(k)}-p\left(\boldsymbol{\gamma}^{(k)}\right)+y_1\\ \vdots\\ \nabla p\left(\boldsymbol{\gamma}^{(k)}\right)\boldsymbol{\gamma}^{(k)}-p\left(\boldsymbol{\gamma}^{(k)}\right)+y_T\end{bmatrix}. \tag{5-12}$$

记：

$$\phi_t(\boldsymbol{\gamma})=p(\boldsymbol{\gamma})-y_t, \tag{5-13}$$

$$\boldsymbol{\phi}=\begin{bmatrix}\phi_1(\boldsymbol{\gamma})\\ \vdots\\ \phi_T(\boldsymbol{\gamma})\end{bmatrix}, \tag{5-14}$$

$$\Phi(\boldsymbol{\gamma})=(A_k\boldsymbol{\gamma}-\boldsymbol{b})^T(A_k\boldsymbol{\gamma}-\boldsymbol{b}). \tag{5-15}$$

原问题（5-9）即可化为：

$$\min_{\alpha,\beta}\sum_{t=1}^{T}\Phi(\boldsymbol{\gamma}). \tag{5-16}$$

直接求解目标函数（5-16）梯度为 0 的点，即目标函数在$\boldsymbol{\gamma}^{(k)}$处的最小值点：

$$A_k^TA_k\left(\gamma-\boldsymbol{\gamma}^{(k)}\right)=-A_k^T\boldsymbol{\phi}. \tag{5-17}$$

由于$A_k^TA_k$不是列满秩矩阵，考虑使用 Levenberg–Marquardt 算法[35]使用$A_k^TA_k+\mu\boldsymbol{I}$ 代替 $A_k^TA_k$. μ为 LM 算法的惩罚因子，于是解得：

$$\boldsymbol{\gamma}^{(k+1)}=-(A_k^TA_k+\mu\boldsymbol{I})^{-1}A_k^T\boldsymbol{\phi}+\boldsymbol{\gamma}^{(k)}. \tag{5-18}$$

记：

$$h_{lm}=-(A_k^TA_k+\mu\boldsymbol{I})^{-1}A_k^T\boldsymbol{\phi}. \tag{5-19}$$

所以获取最小化目标函数的参数 $\boldsymbol{\gamma}=\{\alpha,\beta\}$ 的方法：

输入：初始值α, β, 收敛误差ϵ ，令 k =0

1. 选取μ, 计算$h_{lm}=-(A_k^TA_k+\mu\boldsymbol{I})^{-1}A_k^T\boldsymbol{\phi}$
2. 计算$\boldsymbol{\gamma}^{(k+1)}$=$\boldsymbol{\gamma}^{(k)}-h_{lm}$
3. 计算$p^{(k+1)}\left(\boldsymbol{\gamma}^{(k+1)}\right)-p^{(k)}\left(\boldsymbol{\gamma}^{(k)}\right)$与$\epsilon$比较，大于则回到 1，小于则结束

μ 的选择采用信赖域方法求解[35]：

定义

$$\Delta P^{(k)}=p^{(k+1)}\left(\boldsymbol{\gamma}^{(k+1)}\right)-p^{(k)}\left(\boldsymbol{\gamma}^{(k)}\right), \tag{5-20}$$

$$L^{(k)}(h)=\nabla p\left(\boldsymbol{\gamma}^{(\boldsymbol{k})}\right)h_{lm}+\frac{1}{2}h_{lm}^TB^{(k)}h_{lm}. \tag{5-21}$$

其中$B^{(k)}$为$\nabla^2p(\boldsymbol{\gamma})$的第$k$次近似。

$\Delta L_k=L^{(k)}(0)-L_k(h_k)s$

即通过监控步长质量：

$$q_k=\frac{\Delta P_k}{\Delta L_k}. \tag{5-22}$$

初始时μ_0选择尽量小，如果步长$q_k<0$，那么$\Delta P_k<0$，迭代点失效，需要缩小信赖半径μ重新求解。q_k接近 1，新迭代点效果较好可以继续，下一次可以增加信赖半径μ，其余情况保持不变。

在拟合了 Gamma 函数之后，考虑按照一定的时间间隔，维护每一条在模型

中参与计算的文章情感得分自适应淘汰变量$\zeta_i, 0 < i \leq N$。由于淘汰变量的数学定义，使得淘汰变量本身服从正态分布，越靠正态曲线左侧的数据在模型中越不重要，越靠正态曲线右侧的的数据在模型中越重要。根据 Gamma 函数的特性，随着时间的推进，一条数据在模型中贡献会快速由普通到最大，然后渐渐不重要。

为了得到淘汰变量决定的具体的淘汰策略，首先通过极大似然估计法[36]，计算出正态分布的参数估计：

$$\hat{\mu} = \frac{\sum_1^N \zeta_i}{N}, \tag{5-23}$$

$$\widehat{\sigma^2} = \frac{(\zeta_i - \hat{\mu})^2}{N-1}. \tag{5-24}$$

记：

$$\widehat{\Phi}(x) = \Phi\left(\frac{x-\hat{\mu}}{\widehat{\sigma^2}}\right). \tag{5-25}$$

定义数据最大容量：

$$V = kv. \tag{5-26}$$

其中v为当前话题的默认允许的文章数量，有效时间为τ，选取淘汰阈值为m，数据池为S.

按照如下规则选择进行更新：

1. 每一次时间窗口，将模型中超过时间τ的文章数据淘汰出有效推特，重新计算有效推特数量v，重新计算一次参数估计$\hat{\mu}$, $\widehat{\sigma^2}$，
2. 每一次时间窗口，$\zeta_i < \Phi(m)$的数据进行模型淘汰
3. 每一次时间窗口，将新来数据加入数据池，淘汰数据池中超过时间 τ 的已经过期的数据。将数据池中$\zeta > \Phi(m)$的数据会加入模型更新，没有加入模型的数据重新加入数据池，如果此时模型中的数据大于 V，则将模型中的按照ζ_i从小到大的顺序进行末位模型淘汰。

结束

通过调整 k 的大小可以调整参与影响文章的数量，k 取比较大时，其参与影响模型的总数量会比较多，k 取比较小时，每次末位淘汰的数据会比较多，会影响使得μ的向正态曲线右侧移动，会使得加入的数据要求更严格。所以 k 比较小时重新计算的复杂度会降低。

通过调整τ的大小可以调整不同话题热度的变化速度，τ的取值一般应该与 Gamma 函数的β保持相对一致。可以取τ与β保持线性的关系。

通过调整 m 的大小可以调整加入模型数据的准入阈值，阈值较大时对于模型的计算量会小，参与计算的数据的情感倾向会普遍比较明显，偏向中立的情感评价加入的概率会变小。阈值较小时，参与计算的数据计算量会变多，计算数据的情感倾向对于整个平台而言更有代表性。

每一次时间窗口在完成更新之后会计算出每一篇文章数据的情感得分，通过计算其社交影响力的加权平均值，可以计算出该话题对应的上市公司最终的情感评分。

5.3 Passive Aggressive SVM 算法的设计

利用动态淘汰策略的在线 SVM 算法，可以自适应地随着新到来数据调整模型，而不用重新计算所有数据。但是该算法存在由于参数设置问题导致模型更新过快，旧的数据尚未产生其足够的舆论影响即被淘汰的情况。该算法也会出现历史数据积累的效应无法在时间上对模型产生完整影响的情况。在此基础上，本文提出使用一种新的在线被动攻击 SVM 算法，通过对目标函数进行变形，从而能满足在一定条件下最小化每一条数据对模型产生的影响，且历史数据的影响不被快速覆盖。

将当前时刻现有所有的对于该对象的语言评价的情感得分以及该时间内新到来的的某一条或多条评价进行组合定义，即当前时刻所有的评价矩阵

$$W_n = [w_n^T, w_c^T], W_n \in R^{\{(m_1+\ m_2)*\ k\}}. \tag{5-27}$$

为了达到满足一定条件下最小化当前时刻每一条数据对模型产生的影响的效果，即最小化W_t，可以认为找到最小化的目标函数

$$min\left(\frac{1}{2}\|W\|_t^2 + \frac{1}{2}C\sum \xi_k\right),$$

$$s.t.\left\|S_p W_n\right\|_2^2 - \|S_n W_n\|_2^2 \geq \delta(S_p, S_n) - \xi_i, \qquad \forall\ i. \tag{5-28}$$

其中，ξ_k为模型中允许犯错的松弛变量，S_p为被允许加入模型数据，S_n为被拒绝加入模型或者从模型中淘汰的数据。

对于此类目标函数，可以使用在线被动攻击算法[37](Online Passive Aggressive Algorithms, Crammer)变形为迭代方式优化，即每一步需找到

$$Q_{t+1} = \frac{1}{2}\|Q - Q_t\|^2 + \ \ C\ \ \xi_t,$$

$$s.t.\ \ S_p Q S_p^T - \ \ S_n Q S_n^T \geq \delta(S_p, S_n) - \xi_t, \qquad \forall i. \tag{5-29}$$

其中

$$Q = WW^T. \tag{5-30}$$

目标方程是为了保证Q_{t+1}能够尽可能的高效分类，同时Q_{t+1}应当尽可能和Q_t保持相近，以便能保留先前步的结果。目标方程的拉格朗日函数为：

$$L(Q, \xi) = \frac{1}{2}\|Q - Q\|_t^2 + C\xi_t + \beta\left(S_n Q S_n^T - S_p Q S_p^T + \delta(S_p, S_n) - \xi_t\right). \tag{5-31}$$

其中β为函数的拉格朗日乘子，求出 L 对于 Q, ξ的偏微分为 0 的点：

不妨令：

$$G = S_n S_n^T - S_p S_p^T, \tag{5-32}$$

$$\frac{\partial L(Q, \xi)}{\partial Q} = Q - Q_t + \beta G = 0. \tag{5-33}$$

得到：

$$Q = Q_t - \beta G. \tag{5-34}$$

并且有：

$$\frac{\partial L(Q,\xi)}{\partial\ \xi} = C\xi_t - \beta = 0 \tag{5-35}$$

得到

$$\xi_t = \frac{\beta}{C}. \tag{5-36}$$

重新代入方程，可以得到：

$$\begin{aligned} L(Q,\xi) &= \frac{1}{2}\|Q_t + \beta G - Q_t\|^2 + \frac{1}{2}\left(\frac{\beta}{C}\right)^2 + \beta(\delta(S_p,S_n) - \frac{\beta}{C} - S_n S_n^T + S_p S_p^T \\ &= \frac{1}{2}\|\beta G\|^2 - \frac{\beta^2}{2C} + \beta\left[\delta(S_p,S_n) - \left(S_n Q S_n^T - S_p Q S_p^T\right)\right] \\ &= 0. \end{aligned} \tag{5-37}$$

令β不为 0，常数 C 替换为$\frac{1}{C}$，可以得到：

$$\beta = \frac{X + \delta(S_p,S_n)}{2Y - \|G\|^2 + C}, \tag{5-38}$$

其中

$$X = S_n G S_n^T - S_p G S_p^T, \tag{5-39}$$

$$Y = S_n Q_t S_n^T - S_p Q_t S_p^T. \tag{5-40}$$

于是可以得到算法更新的过程如下：

最小化模型更新的 Online Passive Aggressive SVM
输入：用户 *U* 所有已有推文对于对象 *O* 的评价矩阵 *W*，新到来的评价S_p，未加入或者被淘汰的评价 S_n，参数 *C*
输出：*Q*
1. 初始化：Q_1为一个半正定矩阵，不妨假设为单位矩阵 2. for t=1, 2, 3, … 计算$X = S_n G S_n^T - S_p G S_p^T$ 计算$Y = S_n Q_t S_n^T - S_p Q_t S_p^T$ 计算$\beta = \frac{X\ +\delta(S_p,S_n)}{2Y-\|G\|^2+C}$ 计算$Q_t = Q_t - \beta G$,其中$G = S_n S_n^T - S_p S_p^T$
结束

5.4 基于时间序列的股价分析算法

由于股价在一定程度上服从随机游走模型，其均值来源于市场对于该上市公司的表现预估，方差基于市场的波动情况，在短时间内不考虑非理性因素的情况下，可以认为股价服从随机分布。一般采取时间序列模型[23]对股价进行预测。而时间序列模型一般较多地采用指数平滑法。由于一般可以认为在短期时间内的正常情况下，股价的自然变化没有明显的趋势和倾向，只有在舆论评价产生显著变化的情况下才会对股价变化产生影响，故本文采取一阶指数平滑作为基本模型，

并引入舆论评价因素作为影响股价的变量。

假设上一周期的股价为：P_t,上一周期的预测值为P_t'，舆论评价造成的影响为$\widehat{P_t}$，那么第 t+1 周期其一阶指数平滑的预测结果为：

$$P_{t+1} = aP_t + (1-a-b)P_t' + b\ \widehat{P_t}. \tag{5-41}$$

考虑到股价不可能无限下跌也不可能无限增涨。而由于时间窗口的选择往往远小于舆论传播以及发展的时间。所以相对于预测窗口的时间长度，某一对象的舆论评价可能处于长期持续上扬或者低迷的状态，因此可能会使得预测结果一直处于单一的上涨或者下跌的状态。市场规律指出，实际的股票市场中，一只股票上涨或者下跌的趋势最后由市场表现决定，一段时间的持久的增长会一定程度减缓增长的速率，所以需要考虑历史增长对于预测结果的影响[7]。

根据股票市场的经验，单一一支发行较久的股票，在短时间内出现超过 100%的涨幅十分罕见，可以定义涨跌的相对前τ日的最高价的比例为归一化变量$\beta \in (-1\ ,1)$. 即：

$$\beta = \frac{P_{\max(t+\tau)} - P_t}{P_{\max(t+\tau)}}. \tag{5-42}$$

一般认为短期的历史涨跌的幅度对于未来的市场预期的影响符合近似 sigmoid 的特性[7]，即：

$$\widehat{P_t} = \frac{kQ}{\log^2\left(\frac{1+\beta}{1-\beta}\right)+1}. \tag{5-43}$$

k 为股价值（10 美元到 1000 美元之间不等）以预测结果之间的线性调和系数，Q 为情感分析结果。

于是可以得到：

$$P_{t+1} = aP_t + (1-a-b)P_t' + \frac{bkQ_{j,t}}{log^2\left(\frac{1+\beta}{1-\beta}\right)+1}, \tag{5-44}$$

其中，$Q_{j,t}$为在时间窗口为 t 的时刻，针对于O_j对象的情感分析结果。算法的更新过程如下：

基于时间序列以及情感分析结果的股价预测算法

输入：针对于每一个评价对象O_j，每一个时间窗口 t 的情感评价$Q_{j,t}$，上一个时间窗口的股价P_t，参数 a,b，调和系数 k，影响的天数τ

输出：股价的预测结果$\widehat{P_{t+1}}$

1. P_0初始化为历史 30 天的平均股价
2. 计算$\beta = \frac{P_{\max(t+\tau)}-P_t}{P_{\max(t+\tau)}}$,若$\beta > 1$（极端情况下认为涨幅和接近 100%相同），取$\beta = 1$。计算当前时刻 t 对于对象$O_j$产生的情感评价而造成的股价波动

$$\widehat{P_t} = \frac{kQ}{\log^2\left(\frac{1+\beta}{1-\beta}\right)+1}$$

3. 得到预测值$P_{t+1} = aP_t + (1-a-b)P_t' + b\widehat{P_t}$
4. 下一时刻令 $t = t+1$，回到 2 继续更新股价

结束

参变量 a, b 的选择有多种可以选择，最开始可选取 $a = b = 1/3$，如果一家公司的股票走势趋于稳健，可考虑增大 a. 如果一家公司在经济活动上比较激进冒进，风评容易出现比较大的变化，可以考虑增大 b. 针对不同的公司，参数的选择不尽相同。

第六章 算法的评估

本章主要阐述了训练数据集的选择，包括不同训练集选择对应的讨论主题、模型、参数选择的考虑结果，最后阐述了不同训练集选择情况下的算法性能的比较。

6.1 数据集来源

由于实验涉及的上市公司较多，故采用社交媒体上多个舆论方向的多个数据集进行了算法的评估。

6.1.1 疑影响股价的舆论情感评价

1. 有关核电站的社交舆论情感评价[38]：数据集共 190 条，涉及到来自于 twitter 的社交网络用户对于核电站的情感评价，对上市能源公司的股价可能会产生一定影响；

2. 有关电子产品的社交舆论情感评价[39]：数据集共 11000 条，涉及到来自于 twitter 的社交网络用户对于 iPad，iPhone，Google App，Android Devices 等数码相关的产品的情感评价，可能会对 Apple，Google 以及其他安卓智能设备厂商的股价产生一定影响；

3. 有关航空公司的社交舆论情感评价[40]：数据集共 14610 条，涉及到来自于 twitter 的社交网络用户对于美国各大航空公司的情感评价，对于各大航空公司的股价可能会产生一定影响；

4. 有关亚马逊产品的社交舆论情感评价[41]：数据集约 1500 条，涉及到来自于亚马逊产品的评论，包括 Kindle 等商品，可能对 Amazon 的股价会产生一定影响。

6.1.2 广泛情感评价

1. 对于天气/气候变化的情感评价[42]：数据集 1000 条，涉及来自于 twitter 的社交网络用户对于天气与气候的情感评价；

2. 对于音乐节的情感评价[43]：数据集 3801 条，涉及来自于 twitter 的社交网络用户对于音乐节的情感评价；

3. 对于不限定话题的情感评价[44]：数据集约 40000 条，涉及多个话题的文本评价，评价分为 love，happiness，enthusiasm，worry，neutral，surprise，sadness，relief，boredom，anger 等多个维度，简单再划分为 positive，negative，neutral 等三个维度。数据集数据较多，特征具有普适性。

6.1.3 数据集的字段

数据集的字段如下：

表 6-1 数据集内容以及字段表

数据集名	数据集内容
核电站（数据集 1）	社交舆论评价文本，每条文本积极/中立/消极占比。
电子产品（数据集 2）	社交舆论评价文本，可信度，积极/消极/中立属性 文本发表时间。
航空公司（数据集 3）	社交舆论评价文本，航空公司名，评价可信度，积极/消极/中立属性，文本发表时间/时区。
亚马逊产品（数据集 4）	社交舆论评价文本，产品评分，生产厂商，文本发表时间。
天气/气候变化（数据集 5）	社交舆论评价文本，每条文本积极/中立/消极占比，可信程度。
音乐节（数据集 6）	社交舆论评价文本，可信度，积极/消极/中立属性 文本发表时间。
不限定话题的数据集（7）	社交舆论评价文本，情感属性，文本内容。

6.1.4 股票数据

历史股票价格数据来自于 NASDAQ 网站[45]以及 NYSE 网站[46]，包括历史日期、当日开盘价、当日收盘价、当日最高价、当日最低价，当日交易量、当日市值等六个字段。

对于实时的股票价格数据，可以通过聚合数据[47]的 Web Service 接口获取。同样包含当前价格、当天开盘价、当天最高价、当天最低价，当前交易量、当前市值等六个字段。

6.2 实验设计与评估

实验设计主要分为社交媒体情感分析实验设计与股价预测实验设计两个部分。实验的评估主要针对不同数据集以及不同参数评估了社交媒体情感分析以及股价预测的准确率、AUC、F1 分数以及方差等指标。

6.2.1 社交媒体舆论情感分析

社交媒体情感分析的部分，涉及的算法的组合主要包括：

6.2.1.1 词袋/PCA+TF-IDF/LR

基于词袋的 PCA 与 TF-IDF 算法，采取 LR 作为分类器。对该算法组合进行评估，验证了其准确率，AUC，以及 F1 分数。

假设分类结果的真阳性，假阳性，真阴性，假阴性分别为：$N_{tp}, N_{fp}, N_{tn}, N_{fn}$. 其计算方法分别为：

$$accuracy = \frac{N_{tp} + N_{tn}}{N_{tp} + N_{fp} + N_{tn} + N_{fn}}, \tag{6-1}$$

$$precision \ = \frac{N_{tp}}{N_{tp} + N_{fp}}, \tag{6-2}$$

$$recall = \frac{N_{tp}}{N_{tp} + N_{fn}}, \tag{6-3}$$

$$f1 \ Score = \frac{2 * recall * precision}{precision + recall}. \tag{6-4}$$

AUC 由 ROC 曲线得到，受篇幅所限 ROC 曲线不一一给出，其部分参数下算法的性能如下表所示：

表 6-2 不同参数下的词袋/PCA/TF-IDF/LR 算法的性能评估数

参数	指标	数据集 1	数据集 2	数据集 3	数据集 7
词袋/PCA	Accuracy	0.6803	0.6861	0.6930	0.6679
	AUC	0.7225	0.7304	0.7289	0.6963
	F1Score	0.6923	0.6754	0.6967	0.6156
词袋（2-Gram）/PCA	Accuracy	0.5790	0.6265	0.6413	0.6213
	AUC	0.6256	0.6852	0.6776	0.6515
	F1Score	0.5719	0.6416	0.6552	0.6414
TF-IDF	Accuracy	0.6681	0.6805	0.6800	0.6994
	AUC	0.6957	0.7550	0.7449	0.7992
	F1Score	0.5834	0.6400	0.6524	0.7041

可以看到，对于不同的数据集其正确率较为集中在 0.65-0.70 之间，F1 分数集中在 0.6-0.7 之间，假定情感评分对于股价影响的相对固定，那么对于股价涨跌的预测准确率大约在 0.65-0.70 之间，对于股价预测类的自然变动较大数据而言，其准确率已经比较可观。

可以看到 AUC 值可以都大于 0.65，说明对于不同参数下的模型，对于不同数据集的预测结果都能保持相对的稳定，说明参数的选择有一定的普适性。

6.2.1.2 CBOW/SkipGram 与 Hierarchical Softmax

CBOW 以及 SkipGram 模型均为 word2vec 算法的模型，区别仅在于输入于输出分别是词汇与词汇所在的上下文的词向量，需要考虑选取的参数主要包括：

1. 初始数据样本的选取，$sample = (Context(w), w)$
2. Huffman 树 T 的构建
3. 学习率ζ的选取

表 6-3 word2vec 的参数选择

参数	模型	Sample 选择	T	ζ
参数 1	SkipGram	初始 100	算法工具默认	0.23
参数 2	SkipGram	初始 200	算法工具默认	0.46
参数 3	CBOW	初始 100	算法工具默认	0.67
参数 4	CBOW	初始 200	算法工具默认	1.34

对于 word2vec 算法，由于数据过少或者评价对象比较单一很容易产生过拟合，所以选取数据集 7（40000 条）作为基础数据集，可以得到算法的多项评估结果。

表 6-4　不同参数下 word2vec 的的性能评估

参数	Accuracy	AUC	F1Score
参数 1	0.7523	0.8459	0.7290
参数 2	0.7821	0.8004	0.7577
参数 3	0.8288	0.7624	0.7952
参数 4	0.7956	0.8113	0.7477

可以看到，通过选择不同的初始采样点以及不同学习率的模型，其准确率均大于 0.75，模型对于情感的预测结果较好。将同一个数据集下的 word2vec 方法与词袋/PCA 方法进行比较，如图 6-1 所示

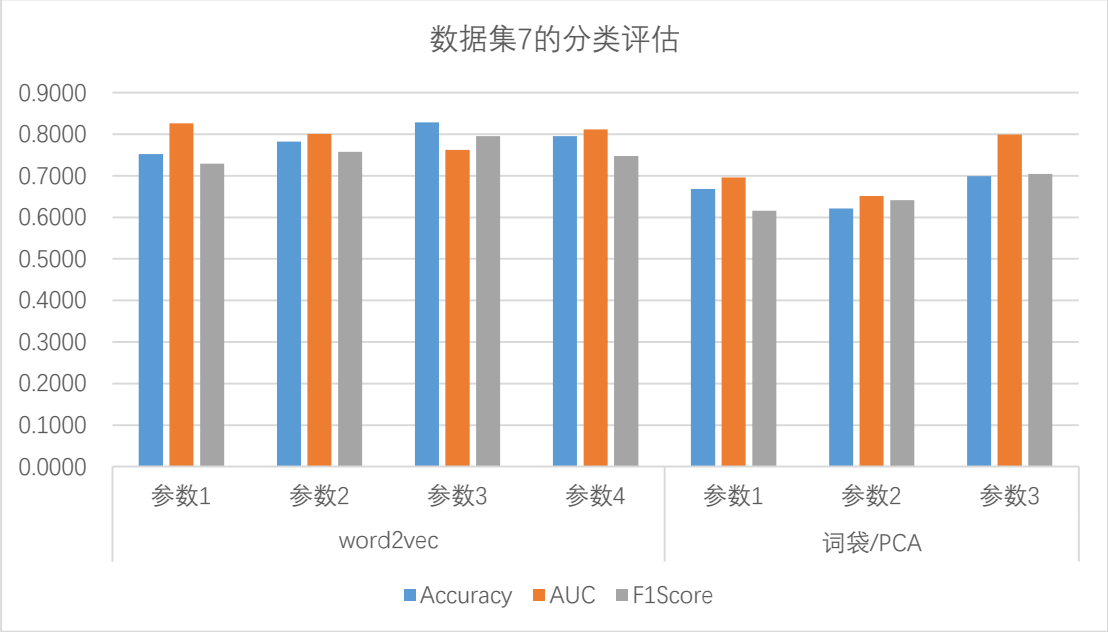

图 6-1　基于数据集 7 的 word2vec 与词袋/PCA 的性能比较

容易看到，对于数据集 7，word2vec 方法无论是 SkipGram 模型还是 CBOW 模型，其整体的准确率，ROC 曲线的指标以及 F1 分数，都要明显优于词袋与 TF/IDF + PCA 模型。这是因为相对于简单的模型对于数据特征的利用十分有限，尤其是数据量比较大的数据集中，简单的模型相对于 word2vec 方法出现了比较明显的欠拟合。但是在本文中，word2vec 以及其利用的分类器有比较明显的缺点：

1. 模型的收敛比较缓慢，与激活函数的选择有很大关系；
2. 模型对于泛化能力不强，同一套参数对于不同数据集会有比较大的波动，这并不满足在线算法的要求；
3. 分类器不是在线算法无法去利用实时数据。

6.2.1.3 Incre-Decre SVM/ Online Passive Aggressive SVM

对于 Online SVM 算法，由于对于模型的更新是实时完成的，历史的舆论可能会影响当前的发帖，所以需要充分利用数据集的文本发表时间这一特性。采用数据集 2、数据集 3、数据集 4、数据集 6 四个有明显时间标签的数据进行实验，选取的参数如下表所示

表 6-5 Gamma 过程拟合的参数选择

参数类型	参数值
Gamma α	0.4
Gamma β	0.6
最大容量比例 k	1.5
有效时间 τ	7(d)
淘汰速度	0.9544

其中α与β以及有效时间τ选取主要考虑到实际社交网络的中一个话题的涟漪效应以及淡忘效应的速度以及影响，最大容量比例选取主要为了防止数据集过大影响计算速度，淘汰速度取了$\Phi(2\sigma)$，认为不满足模型要求的数据处于正态分布的2σ区间之外。

对于模型的评估主要需要考虑随着数据条数的增加其模型的正确率变化，其预测结果的评估可以参考时间序列的评估方法。首先将数据集按照时间窗口划分成多个时间段的数据。根据数据量以及重新计算模型可以接受的时间复杂度（矩阵维度），选取时间窗口为 4 小时，计算 4 小时时间段内的情感评分的平均值并且以时序图表示出来。数据集 4（六家航空公司的情感评价）在时间分布上比较平均，采取数据集 4 进行时序情感评价计算，其结果如图 6-2 所示，其中纵坐标的 1 表示完全消极，3 表示完全中立，5 表示完全积极，0 表示数据缺省。

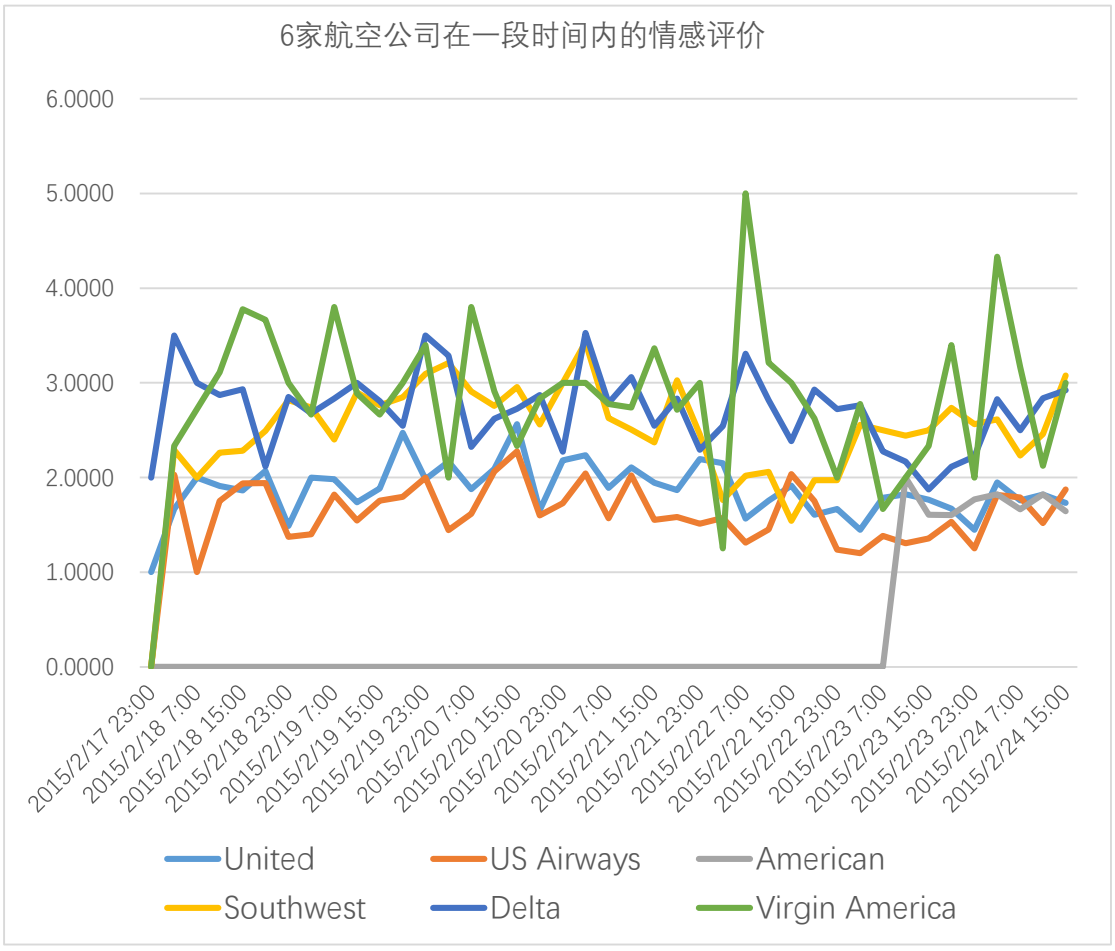

图 6-2 6 家航空公司情感评价在一段时间内的变化

通过分析其自相关程度[35]，六家航空公司情感评价的自相关性如图 6-3 所示。

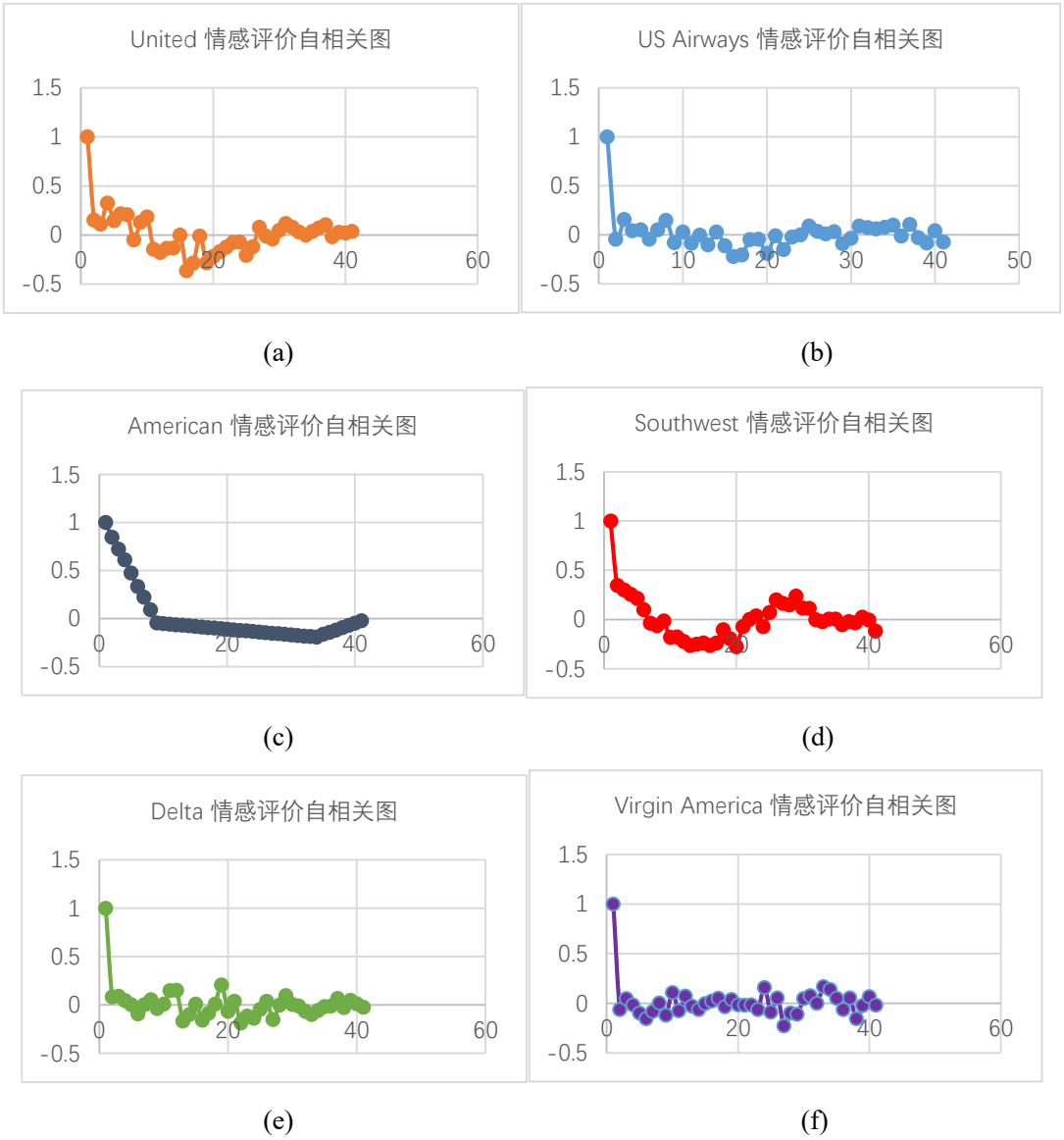

图 6-3　6 家航空公司情感评价自相关性示意图

可以看到，除了 American Airline 由于数据缺失导致前面的自相关性出现明显的直线规律，其余 5 家航空公司的情感评价的自相关系数没有出现截尾或者拖尾，可以认为其情感评价在时间维度上属于平稳白噪声过程。所以对于模型的训练不需要考虑情感评价本身随时间推移的影响[35]。

对于 Incremental-Decremental SVM 的算法，选取单位矩阵作为参数Q_1为进行初始迭代，选取初始推文数为 100 条。

图 6-4 至图 6-9 分别展示了，通过比较预测值以及预估值，基于容量更新的 Incremental/Decremental SVM 算法与 Online Passive Aggressive SVM 的优劣。

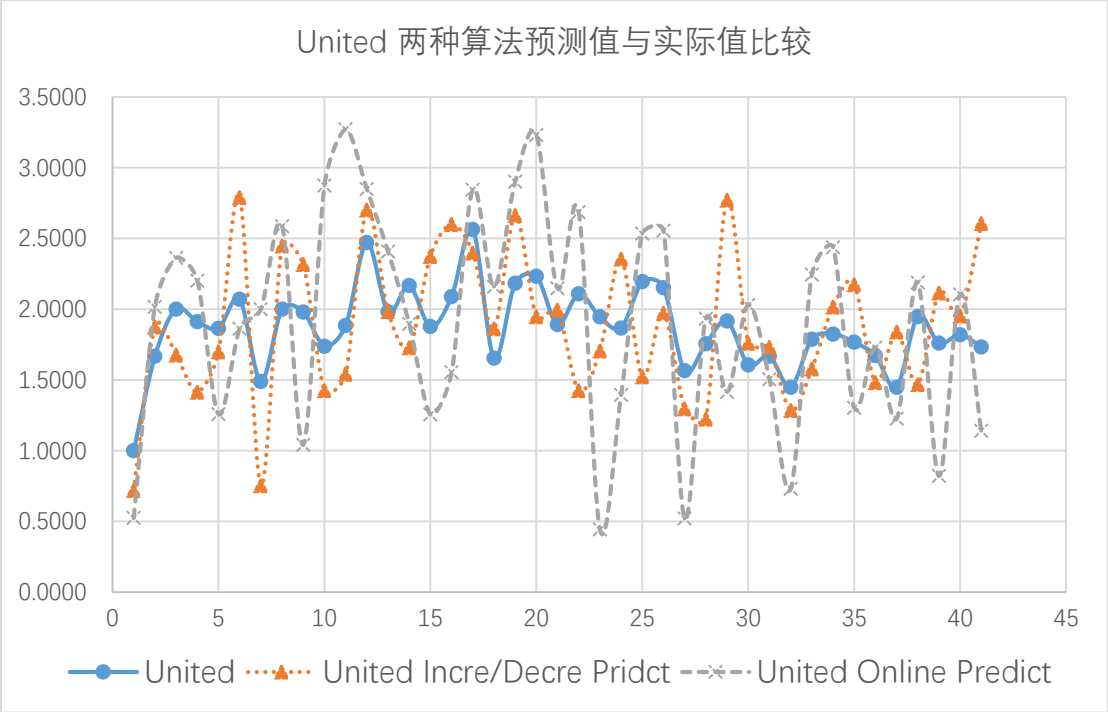

图 6-4 United 航空公司 Online Passive Aggressive SVM 与 Incre/Decre SVM 算法预测值和实际值的比较示意图

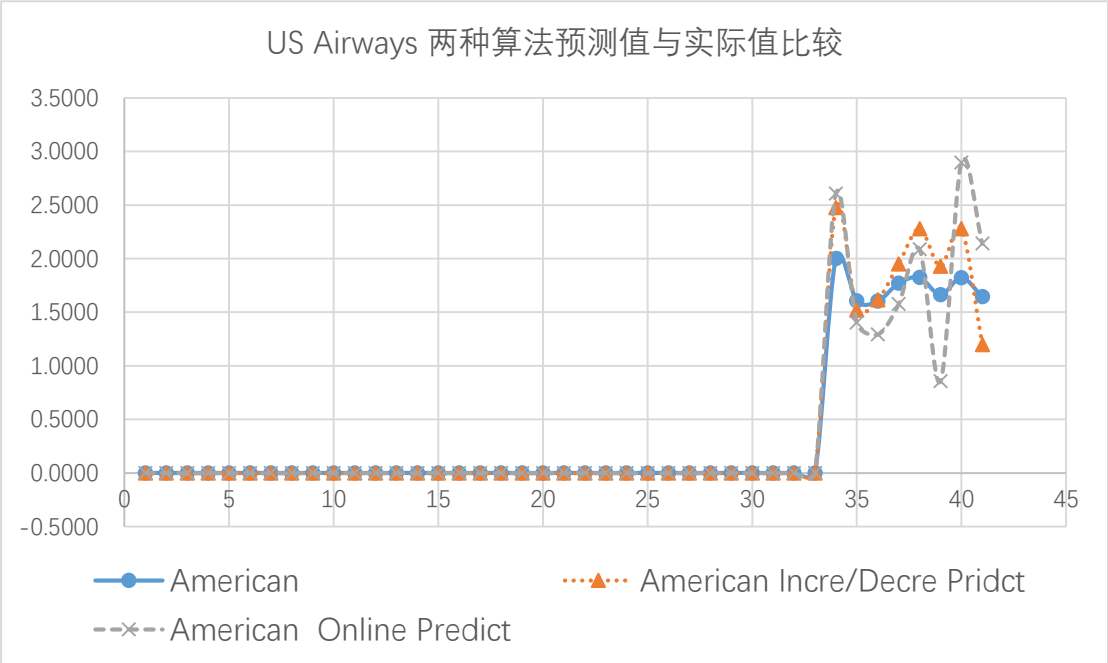

图 6-5 US Airway 航空公司 Online Passive Aggressive SVM 与 Incre/Decre SVM 算法预测值和实际值的比较示意图

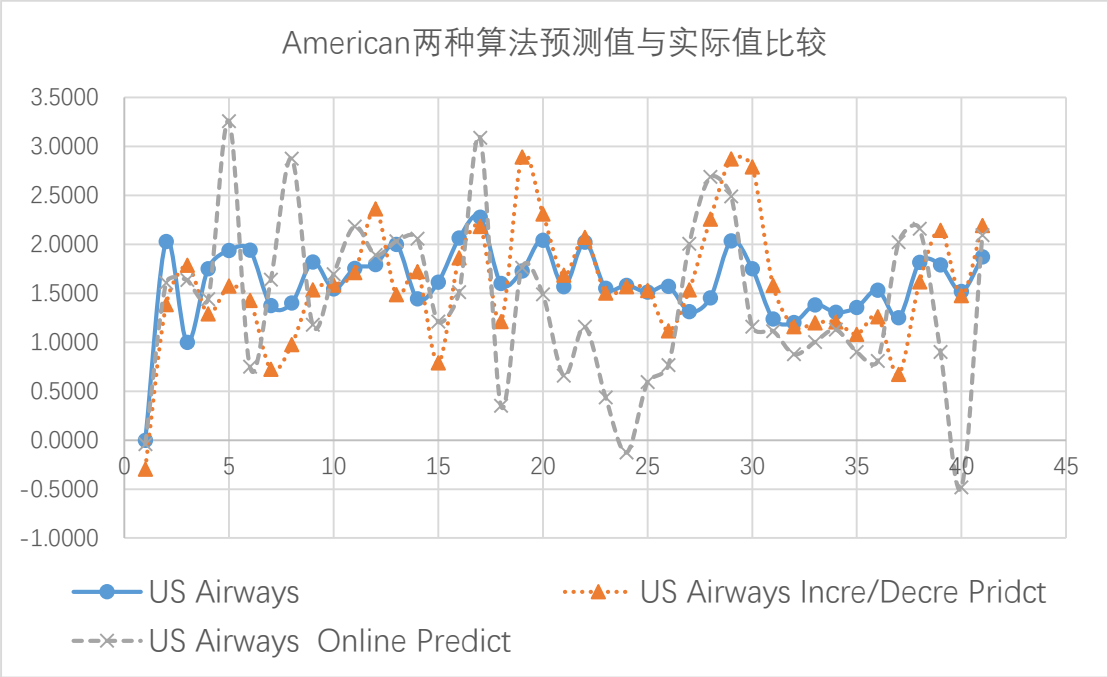

图 6-6 American Airlines 航空公司 Online Passive Aggressive SVM 与 Incre/Decre SVM 算法预测值和实际值的比较示意图

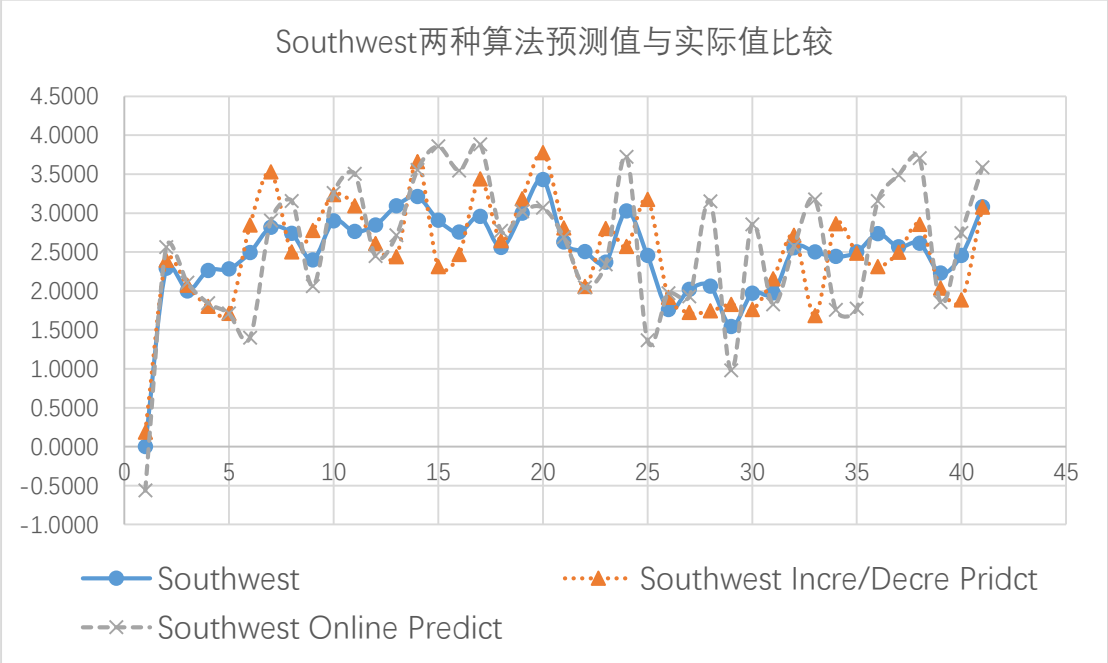

图 6-7 Southwest 航空公司 Online Passive Aggressive SVM 与 Incre/Decre SVM 算法预测值和实际值的比较示意图

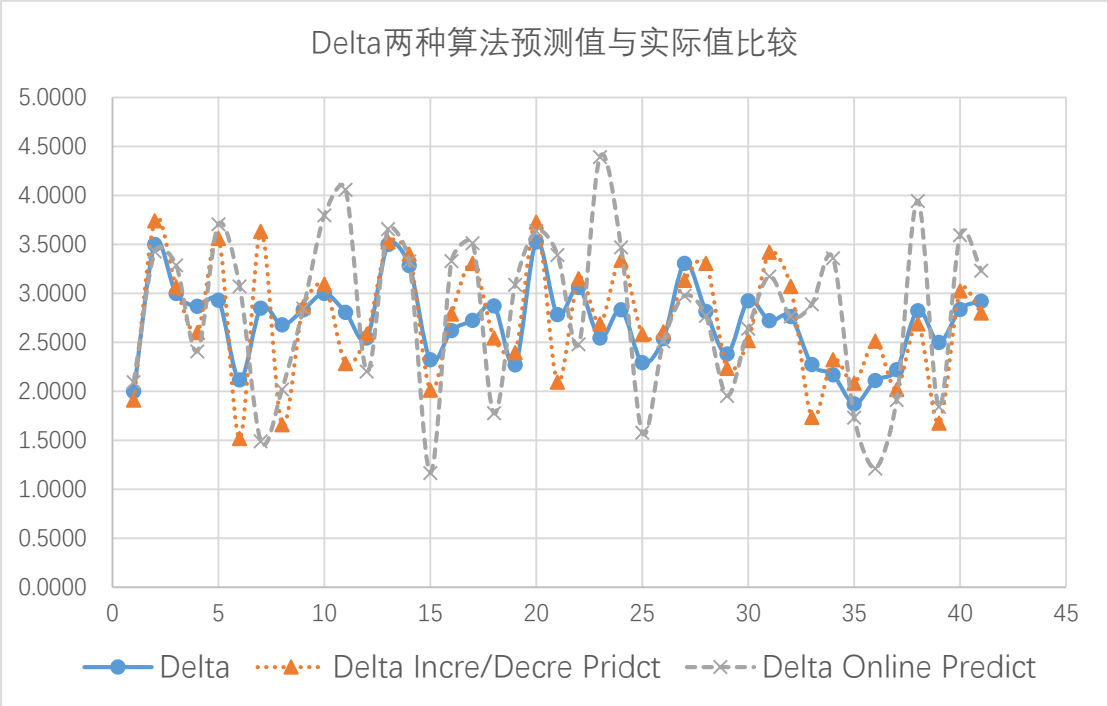

图 6-8 Delta 航空公司 Online Passive Aggressive SVM 与 Incre/Decre SVM 算法预测值和实际值的比较示意图

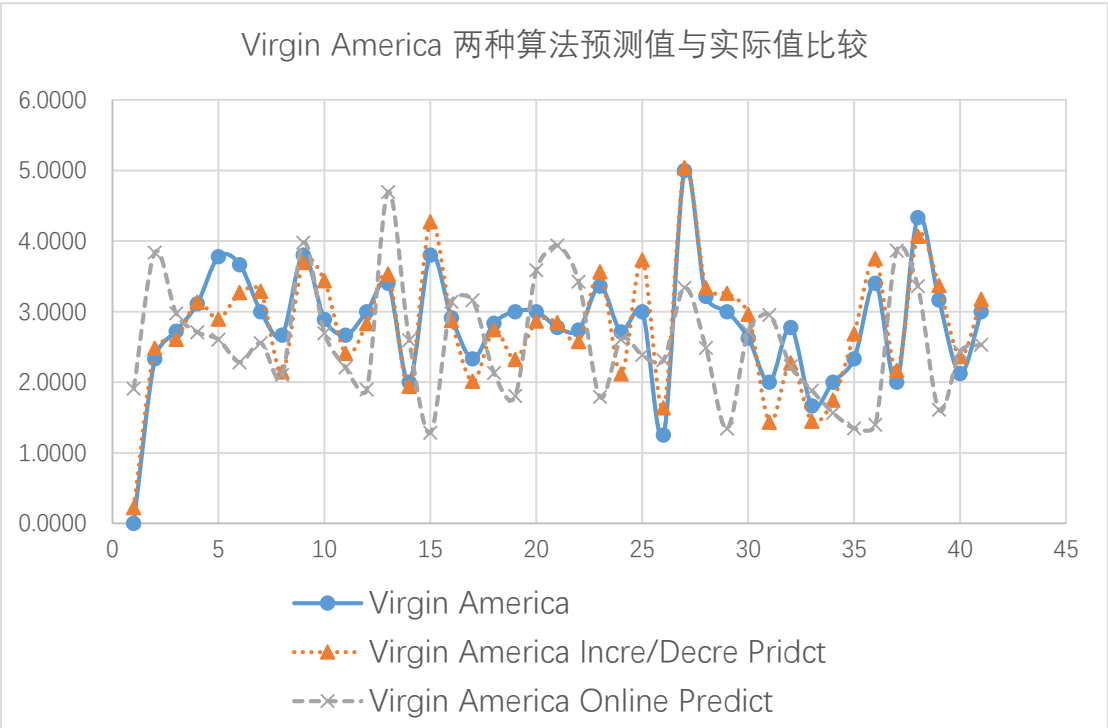

图 6-9 Virgin America 航空公司 Online Passive Aggressive SVM 与 Incre/Decre SVM 算法预测值和实际值的比较示意图

可以看到，两个在线算法的预测效果大致符合了情感的走势，并且根据图像的偏离程度大致可以认为 Online Passive Aggressive SVM 算法在预测准确上要优于 Incremental/Decremental SVM 算法，令第 t 次情感预测值为$\hat{x}_t$，实际值

为x_t，考虑误差：

$$\eta = \sum_t \left(\frac{x_t - \widehat{x_t}}{x_t}\right)^2, \tag{6-5}$$

其算法效果可以总结出以下表格总结得到：

表 6-6　Online Passive Aggressive SVM 与 Incre/Decre SVM 误差指标比较

Airlines	Incre-Decre SVM	Online Passive Aggressive SVM
United	1.720	1.355
US Airways	1.552	1.253
American	6.212	5.821
Delta	2.588	0.889
Virgin	1.778	1.882

根据表 6-6 中 Incremental-Decremental SVM 算法与 Online Passive Aggressive SVM 算法的ζ值比较可以看到，除了 American 二者误差均比较大之外，Online Passive Aggressive SVM 误差整体优于 Incremental-Decremental SVM.

American 航空公司误差相对较大是因为因为数据集中 American 的数据缺失较多，仅有 2015/2/23 11:00 – 2015/2/24 15:00 的部分的 8 个时间窗口。

Online Passive Aggressive SVM 算法优于 Incremental-Decremental SVM 算法是由于，Incremental-Decremental SVM 算法采取了主动淘汰机制，其对于数据的淘汰速度要明显地快于 Online Passive Aggressive SVM，而 Online Passive Aggressive SVM 算法由于最小化了对模型的每一步更新，所以对于新到来数据特征的利用要比 Incremental-Decremental SVM 算法好。但是 Online Passive Aggressive SVM 算法的缺点为：随着时间的增加，其模型包含的数据量会越来越大，导致其算法的复杂度随着运算时间呈线性增长，算法所需要的空间也呈现线性增长。在部署到实际的流处理系统中需要不断进行退化处理以满足实时性的要求。具体每次退化时，需要重新初始化半正定矩阵Q。但是如图上预测可以看出，每次初始化后开始迭代时，算法对于有些航空公司的预测不稳定，这也导致在部署时会存在算法慢启动的问题。

6.2.2 基于时间序列的股价的预测

6.2.2.1 参数的选择

在股价预测部分，设计的算法主要为时间序列算法。不同市场风格公司，例如对于偏向保守的传统能源行业的公司选取参数主要如表 6-7 所示：

表 6-7 时间序列的股价预测的参数选择

参数类型	参数值
调和系数 k	0.43
a	0.6
b	1.5
有效时间 τ	7(d)

6.2.2.2 算法的评估

对于股价的评估，主要评估部署在流处理平台之后的股价预测结果。选取2018/10/11-2018/11/7 日的多个股值的数据，进行跟踪比对。

选取的股指包括：

表 6-8 选取股值的行业分布情况

行业	上市公司
科技行业	Facebook, Amazon, Apple, Tesla, Microsoft, Google, Intel 等 25 家公司
能源行业	Valero Energy, Exxon Mobil, Enterprise Products 等 20 家公司
医药行业	Roche Holding, Pfizer, Novartis, Merck & Co.等 17 家公司
航空行业	United Airline, American Airline, Delta Airline, Southwest Airline 等 12 家公司
娱乐行业	Sony, Activision Blizzard, Walt Disney Co 等 10 家公司

假设每一时间窗口股价的相对变化为 P_t ，预测的变化结果为$\widehat{P_t}$.

选取各个行业部分有代表的公司，其实际股价（黑色与白色圆柱体部分）预测结果（红线标出预测结果）如下方组图所示。

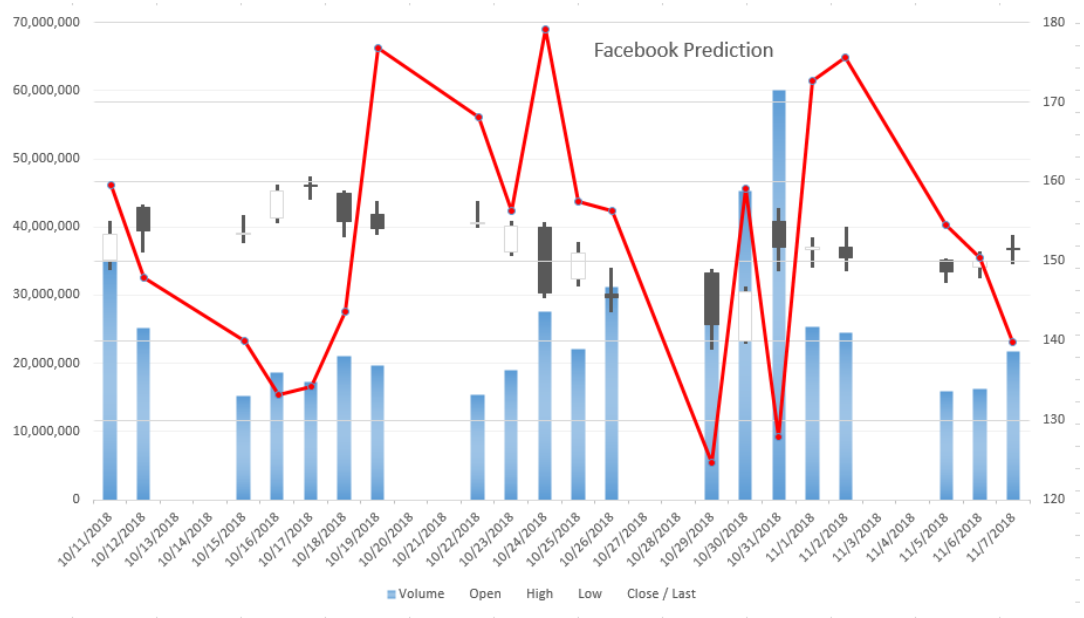

图 6-10 Facebook 20 个交易日股价与预测结果

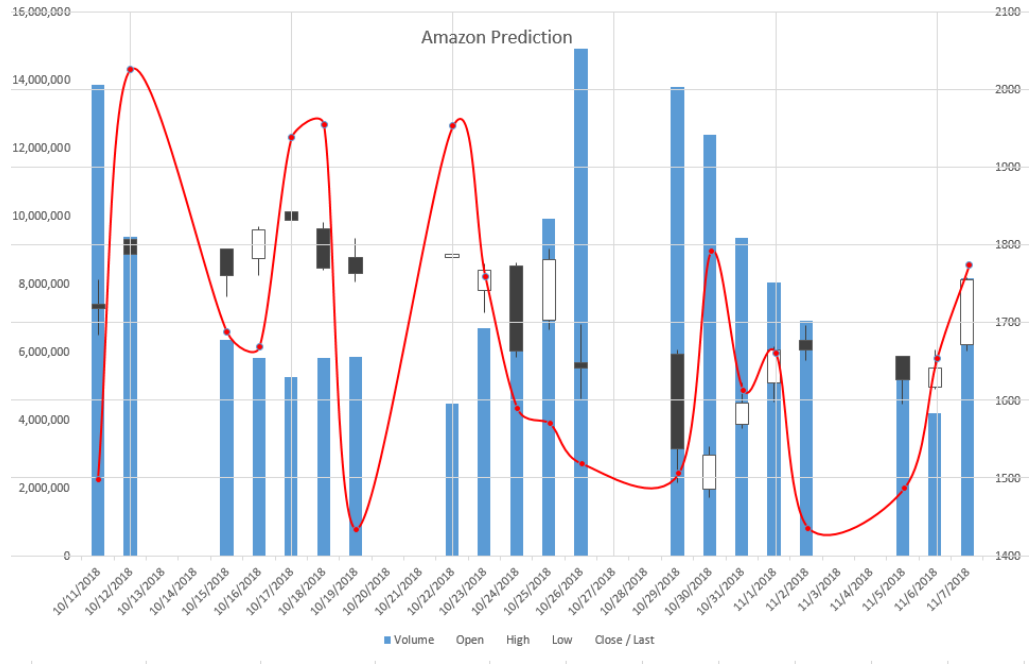

图 6-11　Amazon 20 个交易日股价与预测结果

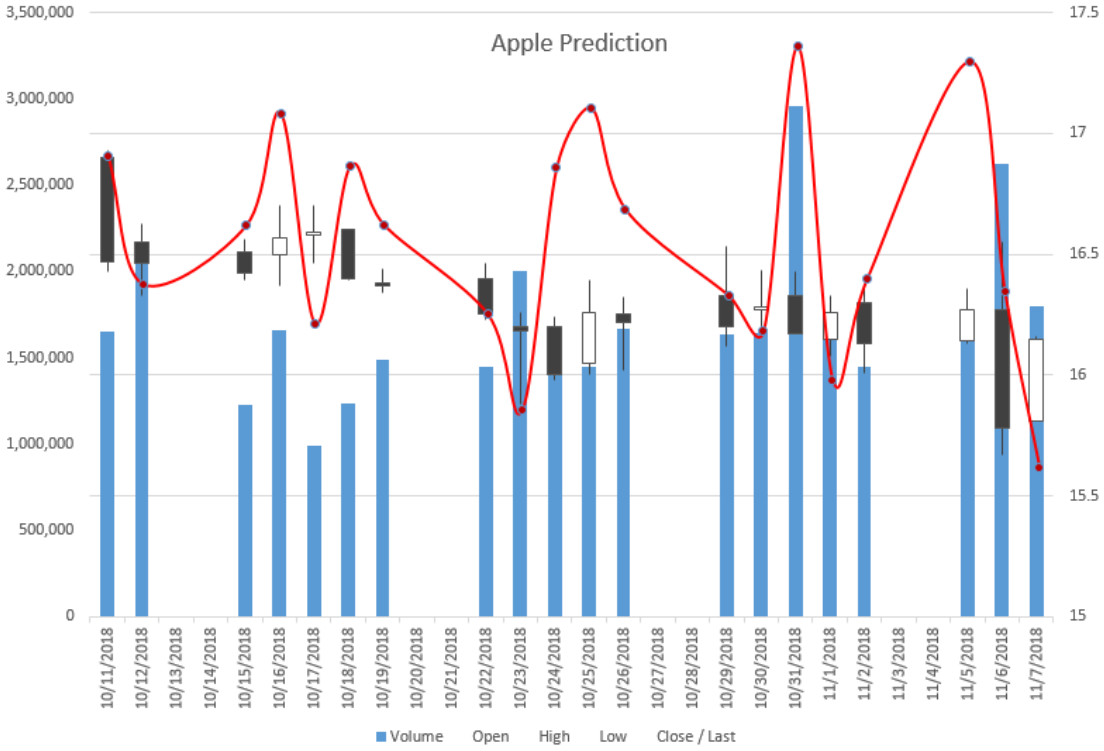

图 6-12　Apple 20 个交易日股价与预测结果

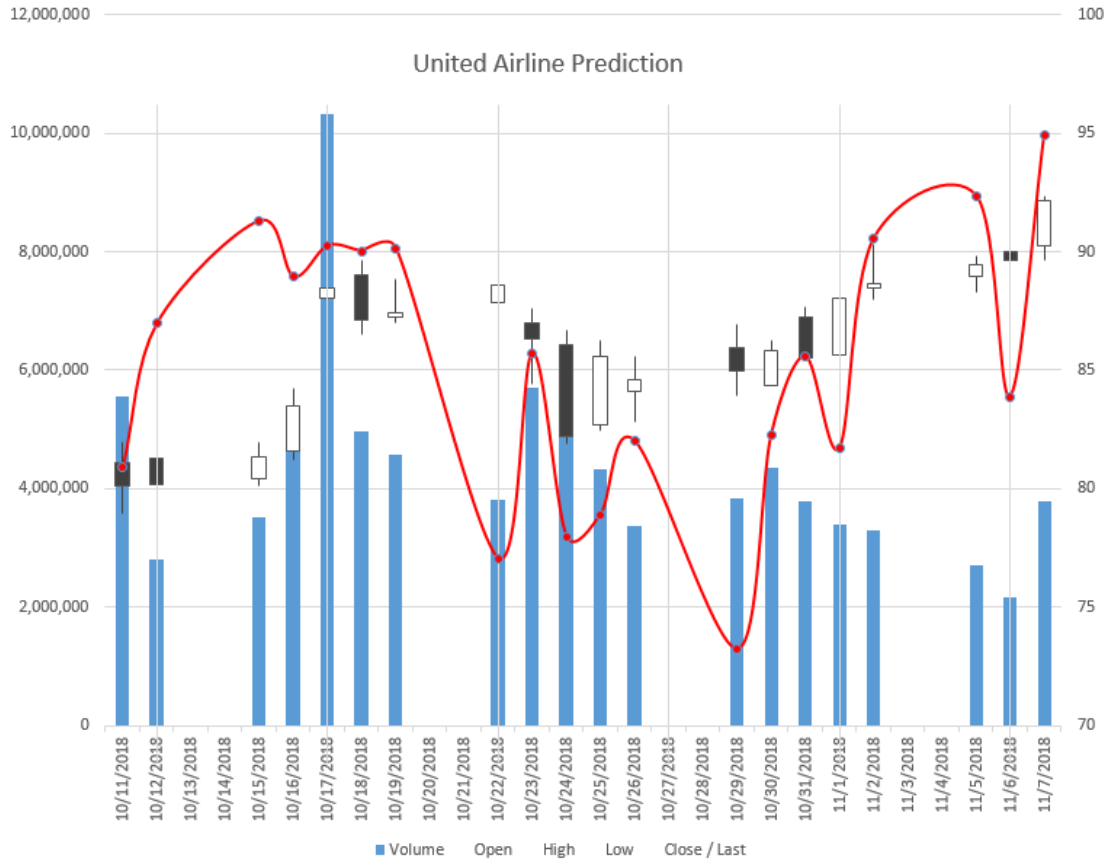

图 6-13 United Airline 20 个交易日股价与预测结果

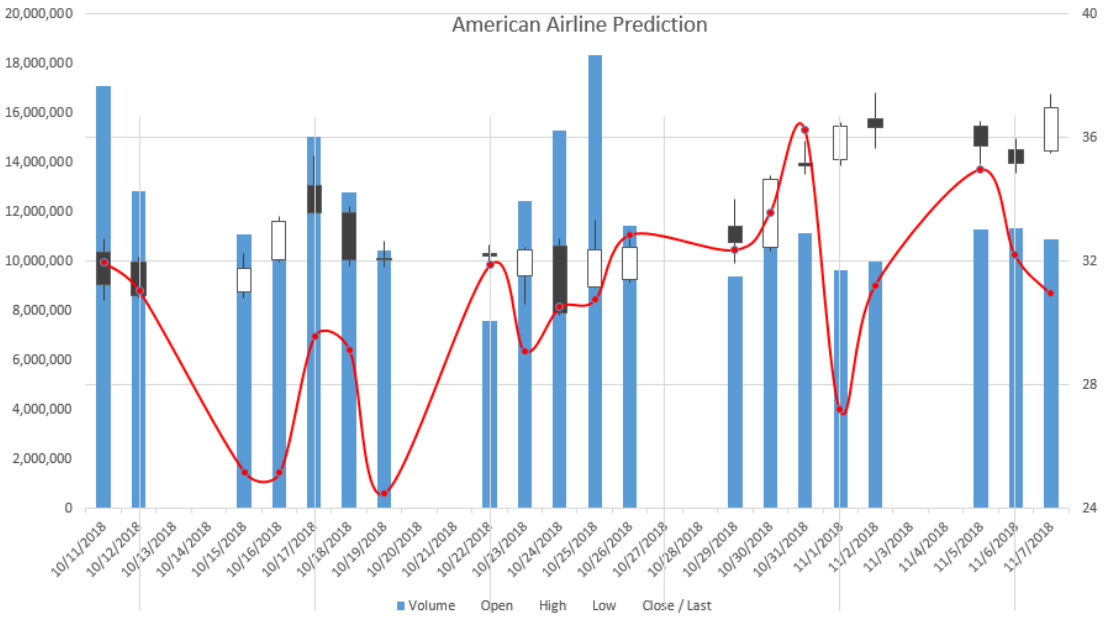

图 6-14 American Airline 20 个交易日股价与预测结果

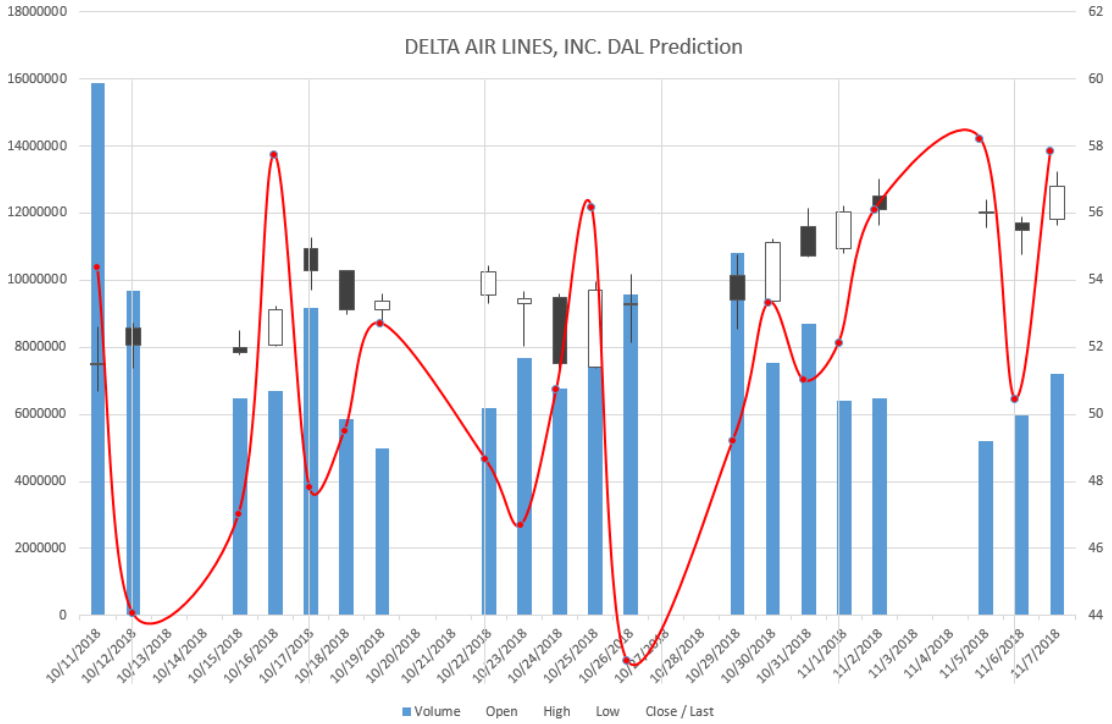

图 6-15　Delta Airline 20 个交易日股价与预测结果

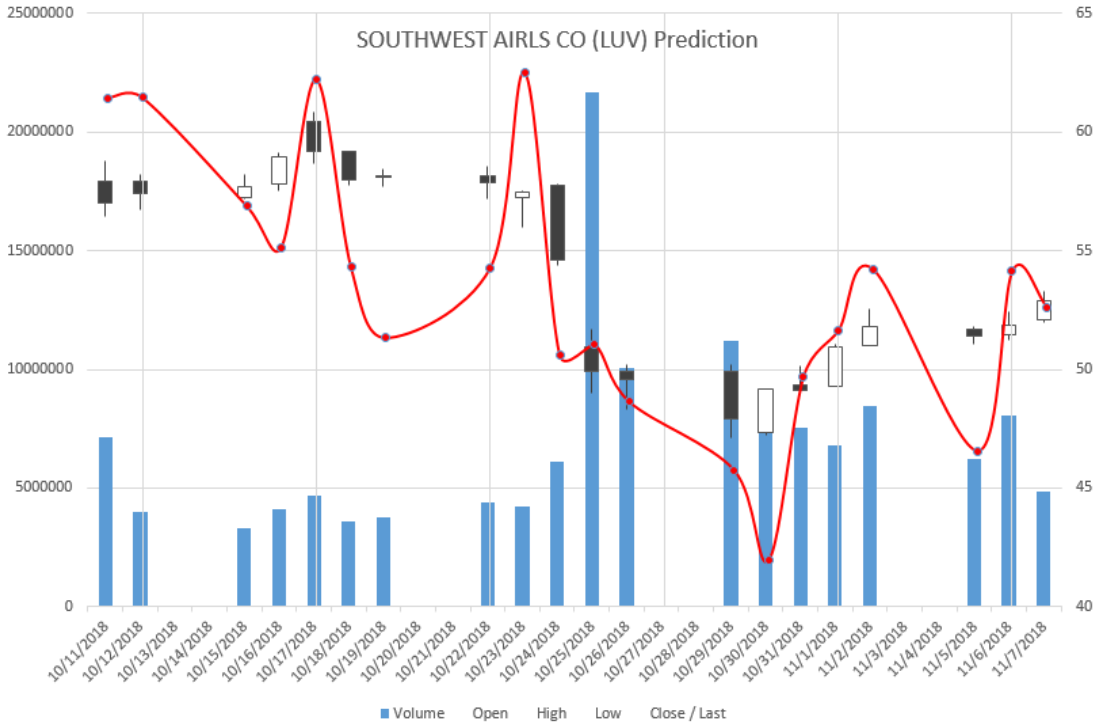

图 6-16　Southwest 20 个交易日股价与预测结果

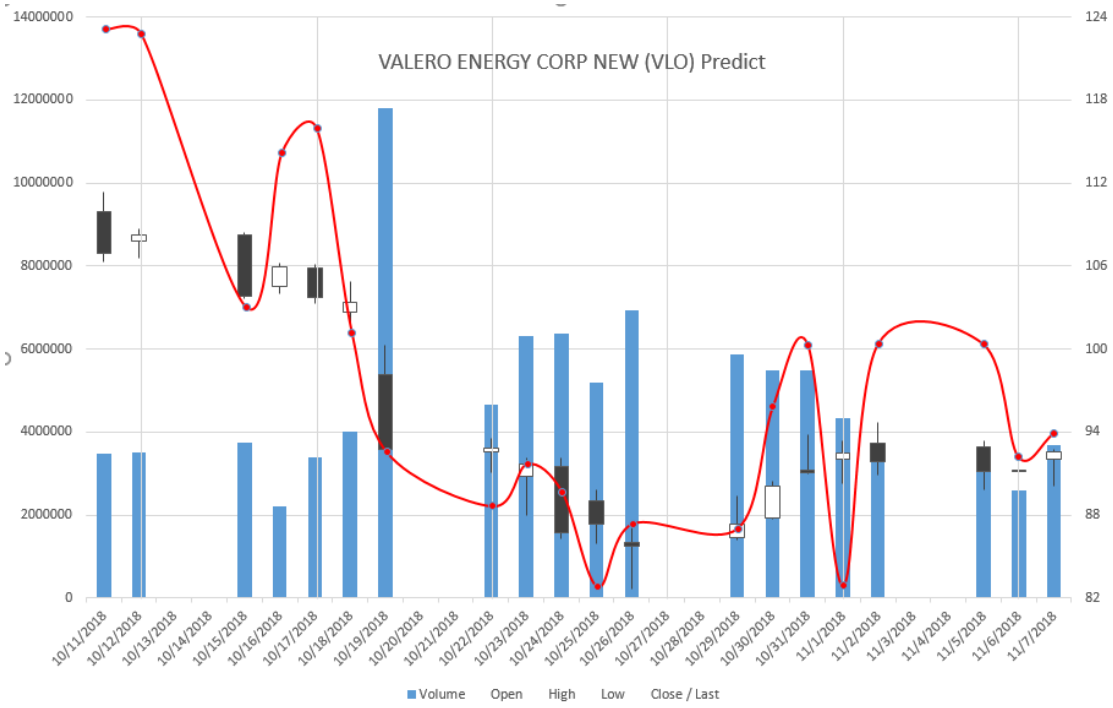

图 6-17　Valero Energy 20 个交易日股价与预测结果

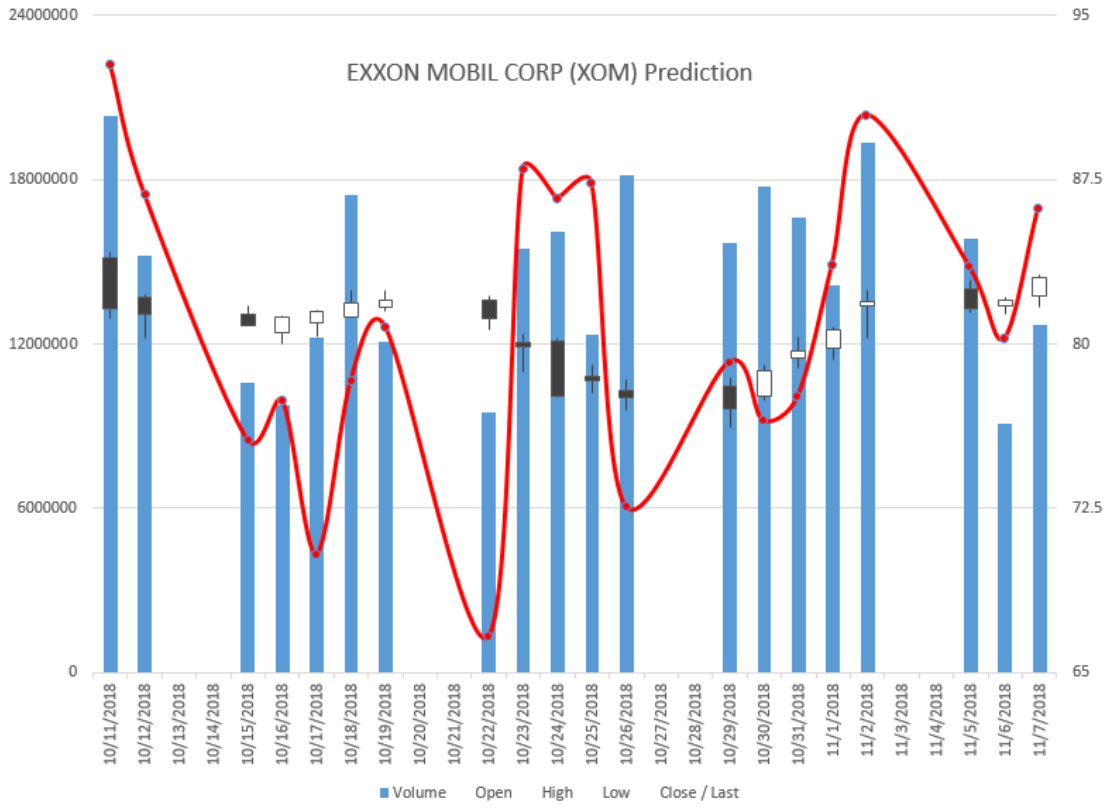

图 6-18　Exxon Mobil 20 个交易日股价与预测结果

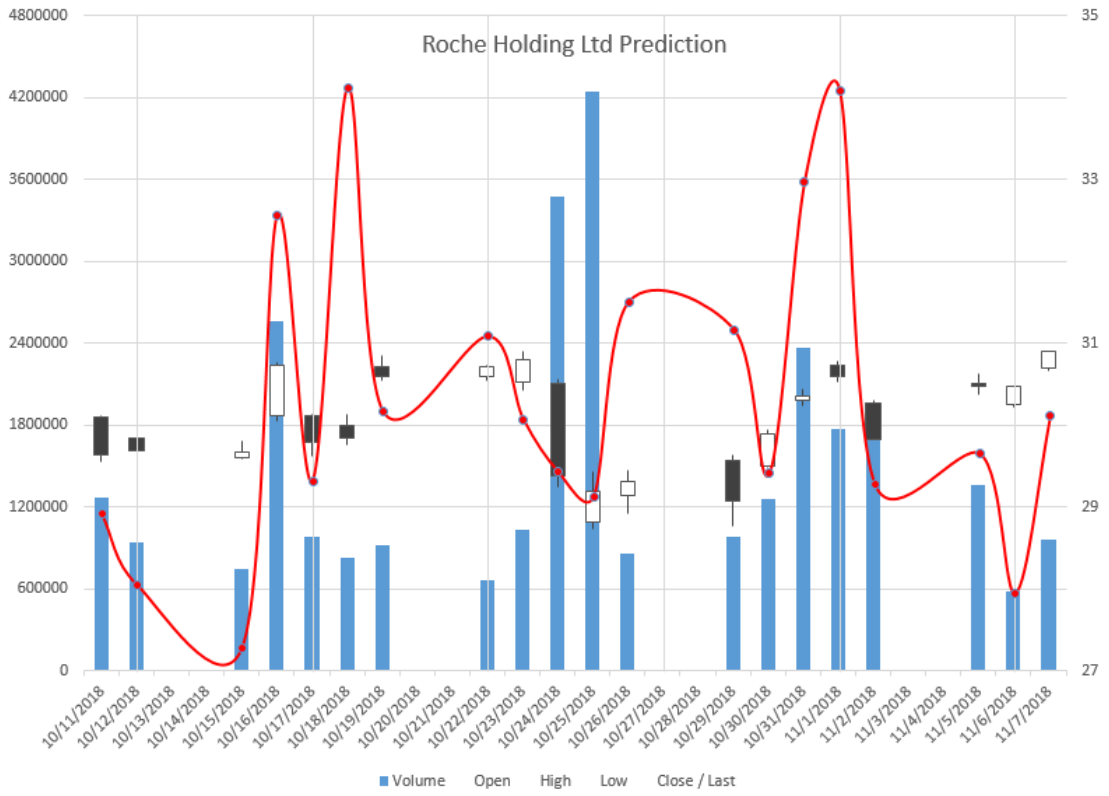

图 6-19　Roche Holding 20 个交易日股价与预测结果

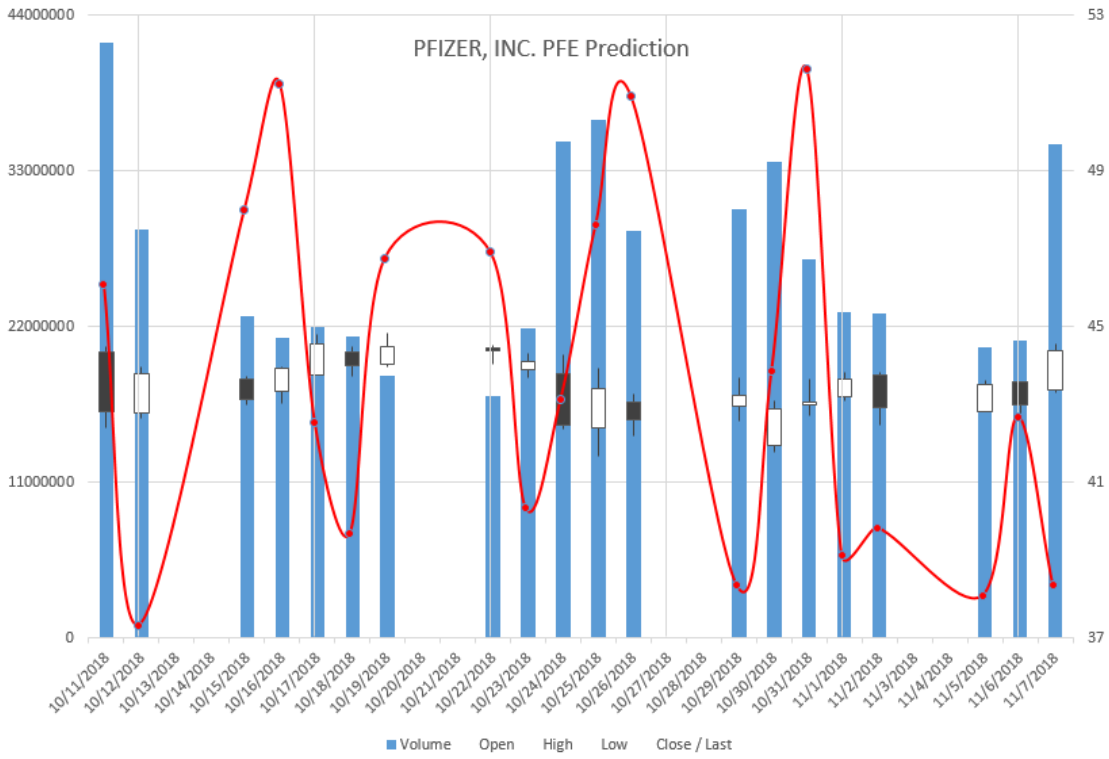

图 6-20　Pfizer 20 个交易日股价与预测结果

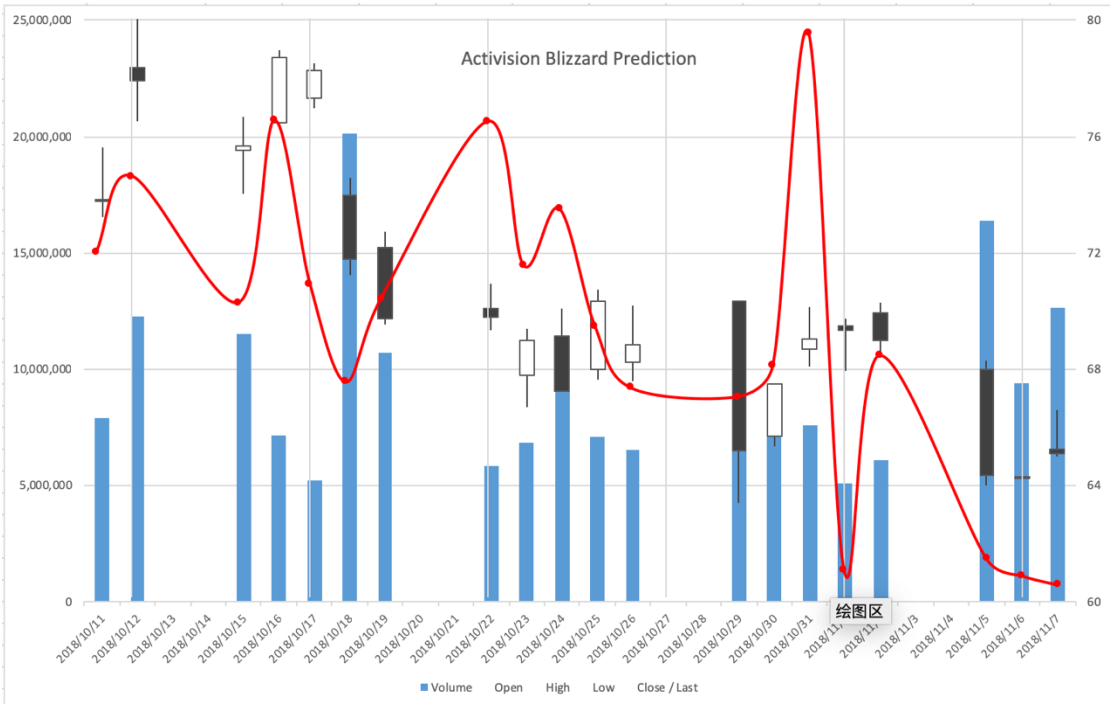

图 6-21 Activision Blizzard 20 个交易日股价与预测结果

考虑时间窗口 t 时刻的预测得分$score_t$以及误差ζ：

$$score_t = 0, P_t * \widehat{P_t} < 0, \\ = 0, P_t * \widehat{P_t} < 0, \tag{6-6}$$

$$Score = \frac{\sum_t score_t}{\sum_t 1}, \tag{6-7}$$

$$\sigma^2 = \sum_t \left(\frac{P_t - \widehat{P_t}}{P_t}\right)^2. \tag{6-8}$$

考虑 2018 年 10 月 11 日-2019 年 1 月 10 日 61 个交易日 182 个时间窗口，可以得到上列组图中不同股指不同算法的预测得分以及误差如下：

表 6-9 不同股值预测得分以及误差情况

公司名	*Score*	σ^2
Facebook	0.685083	2.9802
Amazon	0.563536	3.072
Apple	0.635359	7.8102
United Airline	0.558011	3.8292
American Airline	0.569061	0.864
Delta Airline	0.60221	2.5319
Southwest	0.651934	8.9196
Valero Energy	0.635359	0.7197
Roche Holding	0.651934	2.4914
Exxon Mobil	0.651934	3.2077
Pfizer	0.635359	1.3243
Activision Blizzard	0.662983	3.1396

6.2.2.3 Portfolio 示例选择

得到了股价的预测值后，根据股价预测信息合理分配投资配比（Portfolio）是作为投资参考的最后目的。本文在这里使用一个简单的 Portfolio 示例[48]展示如何根据预测信息配比投资，具体的拓展内容不在本文探讨范围内。

假如考虑股票总投资金额为 W，对于每个上市公司$company_j$考虑投资的分配为w_j,并且假定各个上市公司股价变化的自相关系数$\rho_{i,j}=0$,那么当前投资配比的方差σ^2与收益 S 为：

$$\sigma^2=\sum_j w_j^2\sigma_j^2\,, \tag{6-9}$$

$$S=\sum_j\left(w_j*s_j\right). \tag{6-10}$$

即，投资配比问题为，假定能接受的最大的风险$\sigma^2<risk$，求在该限定条件下的最大化收益 S.

根据拉格朗日乘数法，其拉格朗日函数为

$$L\left(\lambda,w_j\right)=\lambda\left(risk-\sum_j w_j^2\sigma_j^2\right)+\sum_j w_jS_j\,. \tag{6-11}$$

综合风险（误差）与收益（预测得分），可以考虑的分配w_j满足：

$$w_j=\frac{\phi_j\sqrt{risk}}{\sqrt{\sum_j\phi_j}}, \tag{6-12}$$

其中，满足

$$\phi_j=\frac{S}{\sigma_j^2} \tag{6-13}$$

时，股票的可能收益最大。

第七章 基于流处理平台的股价预测系统实现

如第四章系统设计所述，整个基于流处理平台的股价预测系统分为数据获取模块、数据传输模块、数据预处理模块、数据计算模块、数据存储模块以及数据展示模块等六个模块，本章主要根据软件设计规范阐述了六个模块的物理设计以及具体实现过程。

7.1 高并发流量数据获取模块

7.1.1 Tweepy 组件

根据 Twitter 官方提供的 API 文档[49]，用户需要以开放授权（OAuth）的方式，以填写回调 URL 的方式申请 Access Token 与 Access Secret，在完成鉴权（Authentication）后可以使用 OAuth 方式登录 Twitter 账号。然后根据 API 提供的 Web 方法提交 HTTP 请求，从而获取到所需要的基本数据。可能用到的方法包括：

Get Lists/List：得到指定 id（user_id）或者屏幕名（screenname）用户的所有基本信息，包括但不限于用户 id（user_id）、用户全名（full_name）、性别（gender）、关注数（following）、被关注数（follower）等；

Get trends/places：得到指定地区的热词列表，如果获取的热词与预定义的对应上市公司出现匹配，可以认定跟踪的某家上市公司正在被舆论讨论，即该公司股价可能会出现波动；

Statuses/filter：获取当前实时推特的主要方法，通过限定关键词与用户 id 可以追踪到指定用户当前时刻的所有含相关关键词的实时推特。（本方法在系统的调试过程中出现了较大的变更，历史接口为特殊的 twitter stream 接口，仅 GET 方法有 1024 个字节限制，而获取无流量限制，也无需指定用户名，只需指定关键词即可获取当前整个推特网站上所有含关键词的相关推特，而长连接的带宽仅受 TCP 窗口大小的限制，带宽可达到 10M/s，QPS 可达到 10K。调试过程的后期 Twitter 官方开始将该接口改为付费服务，免费版开发者的过滤器限制 5000 名用户与 200 个关键词进行连接，并且数据带宽会被限流。该变更极大地影响了对于实时数据量的要求，后续测试通过注册多个 Twitter 账号暂时缓解该问题，但峰值带宽与 QPS 依然与需求设计有较大差异。）

位于 GitHub 的开源项目 Tweepy 封装了 Twitter 的大部分 Web 接口，使用基于 Python 的 Tweepy 工具可以方便地通过函数调用的方式获取推文，例如使

用 tweepy.Stream(filter)方法可以开启指定过滤器的追踪。

在选择过滤器时，首先根据训练数据时设计到的多个话题方向（包括：科技、医药、能源、娱乐等）选取在 NASDAQ 与 NYSE 上市的 100 余个公司，然后针对每个上市公司选取平均 50 个与该公司有关的关键词，同时追踪北美地区的实时热词，将热词与 5000 余个关键词进行匹配。本系统工作的前期阶段过滤器不需要指定用户，由于后期阶段接口出现较大变更，需要特殊指定用户。考虑到用户的发文习惯，在官方接口调整期对于 MySQL 的 Twitter 全量数据库进行了全库扫描，针对每一个关键词选取了最常评价的一批用户，同时保留了关注、被关注量最多的一批用户，将该批用户的发言进行跟踪，以取代之前对于全站用户的跟踪。

Tweepy 组件（数据获取模块）工作的整体流程过程如图 7-1 所示：

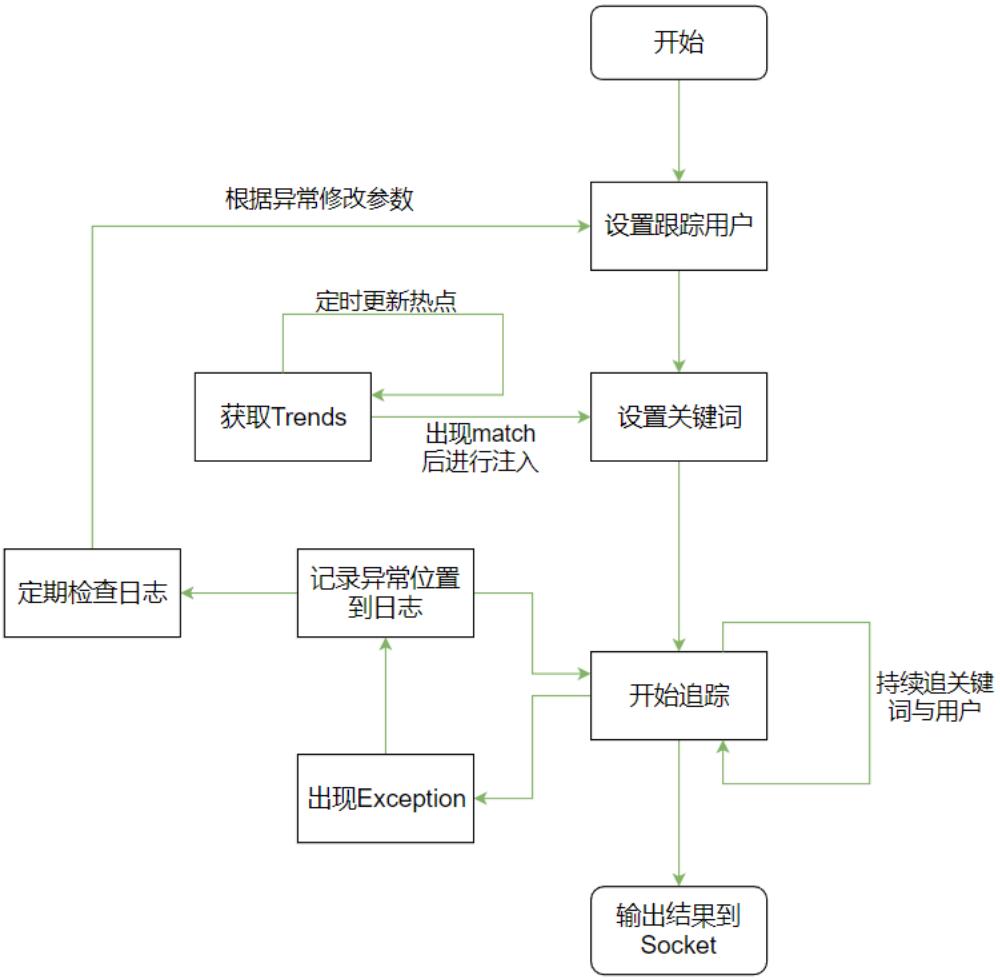

图 7-1 数据获取模块实现示意图

所有的推特数据会直接写入 socket，并在本地进行第一步的预处理，剔除无用字段，仅留下用户关注被关注量、推文 id、推文内容、推文发表时间戳等四个字段。除此之外，推文数据加入了关键词（keyword）字段，方便下游对推特数据按照话题进行分发。经过简单的第一步预处理，每一条推文大小会从

1KB 压缩到 100B.

7.1.2 聚合 API 组件

给定的上市公司的历史股价数据来源与存储主要依靠手动查询网站以及手动导入数据库，数据主要来自于 NASDAQ 的历史数据[45]（history quote），NYSE 的历史数据[46

实时数据来源主要来自于聚合数据的接口。以美股为例，聚合数据[34]提供 Web Service 以 Post 方式返回实时股价的信息。每日的交易时间为 8 小时，交易时间内的当前股价以 HTTP 报文形式进行封装。聚合 API 组件在接收到 POST 报文后直接将当前股价写入数据库。

7.2 高可用数据传输模块

由于推特数据获取模块所在的物理机器架设在海外，所以需要高可用的网络通信框架支持，海外机器才能与架设实验室中心机房的机器进行可靠通信。同时系统还需要利用消息中间件收集数据，从而对产生速度不一的源数据进行缓冲，然后分发到不同的流处理平台进行处理。

7.2.1 Netty 组件

系统选取 Netty 框架作为基础通信框架，发送由数据获取模块简单预处理后的推文。Netty 的 ChannelPipeline 是在传输过程中的一系列责任链，分为 inBound 和 outBound 两个部分。以 inBound 部分为例，包含的传播方法包括：

“ChannelHandlerContext.fireChannelRegistered(),

ChannelHandlerContext.fireChannelActive(),

ChannelHandlerContext.fireChannelRead(Object),

ChannelHandlerContext.fireChannelReadComplete(),

ChannelHandlerContext.fireExceptionCaught(Throwable),

ChannelHandlerContext.fireUserEventTriggered(Object),

ChannelHandlerContext.fireChannelInactive(),

ChannelHandlerContext.fireChannelUnregistered()”

等等。这些方法分别为：重写注册、激活、读取、读取完毕、异常处理、事件触发、下线、注销等方法，而每一个过程对应一个 ChannelHandler.

Netty 的 ChannelPipeline 结构如图 7-2 所示：

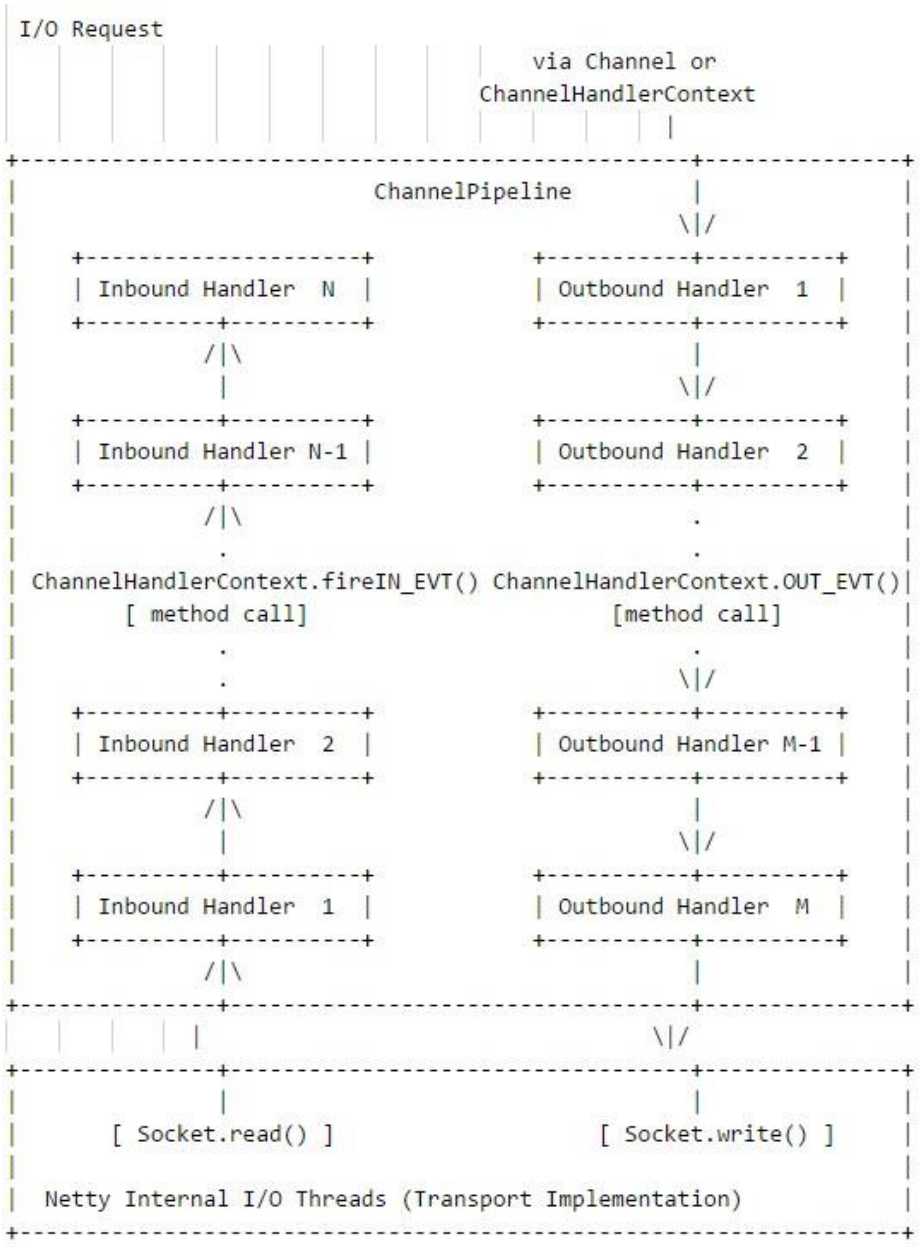

图 7-2　Netty ChannelPipeline 责任链示意图

通过定义各个不同阶段的责任链中的传播方法，定义了一整套通信协议。

Netty 组件的实现需要将写好的 ChannelHandler 注册到 ChannelPipeline 中，然后将 ChannelPipeline 与相应 Group 的 Channel 绑定，Channel 中又绑定了端口号与服务器地址，最后将 Channel 的网络层封装到 socks5 协议里，在本地使用 1080 端口，socks5 协议使用 8088 端口，将报文发送出去。

接收端通过 sslocal 程序接收来自所有 8088 的端口的报文，并转发到本地 1080 端口，Netty Client 端通过重写对应的 Outbound Handler，如图 7-2 所示按照相反的 tail 至 head 的方向完成 OutBound 责任链的传播方法，对报文进行相应 write()，flush()，read()等操作，然后写入内存。

7.2.2 Kafka 组件

Kafka 消息中间件组件首先需要创建对应上市公司的关键词，区分多个 Topic 以便于数据处理模块进行处理。由于对 Netty 的改写基于 Java API，将推文写入数据库也是在 JVM 层面完成，所以采用 Kafka 的 Java API 进行消息中间件的调用，消息队列实现逻辑如图 7-3 所示：

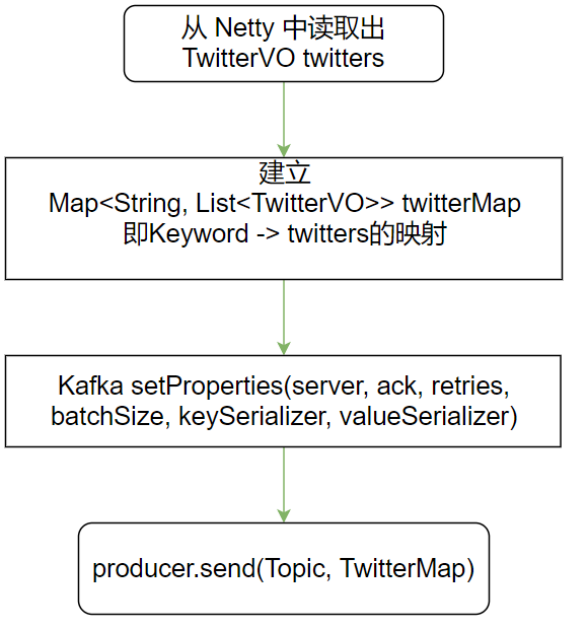

图 7-3 Kafka 消息队列实现示意图

在 Consumer/Broker 端，不同的 Consumer 实例部署在不同的机器上，而不同机器的部署按照处理不同 Topic 的流处理平台进行分配。Broker 代码逻辑与 Consumer 相似。具体机器的分配如表 7-1 所示：

表 7-1 按照话题的机器分配表

Topic	Example	Machine
Tech	Apple, Facebook, Google	10.108.247.103@0
Airline	Companies Delta, American Airlines	10.108.247.103@1
Energy	EXXOL	10.108.247.103@2
Entertainments	Blizzard, Sony	10.108.247.103@3

7.3 Spark Streaming 数据预处理组件

流数据的预处理主要由 Spark Streaming 完成，需要在之前简单的预处理基础上完成进一步的处理，包括：

1. 词汇大小写转换，同样大小字母的单词表达了同样或者相似的含义，可

以处理为同一个 String 对象。除此之外，网络用语还存在一些特殊的字符层的对应法则，也需要进行判断后进行转换，部分转换规则如表 7-2 所示：

表 7-2 字符级别部分转换表

3	e
!	i
4	A
6	b
0	O
∂	d
1	l
2	z

2. Emoji 等特殊词汇转换，由于推特文本的传输格式为 Unicode 模式，每一个字符被 4 个字节编码，emoji 等特殊表情同样也可编码为 4 字节的 Unicode 字符。而社交网络理论认为，表情符号往往有比较明显的情感倾向，所以必须对这些表情进行转换处理。具体做法是，读取 Emoji 的 Unicode 编码，并且按照预设的字典转化为对应的情感表达。例如”u1f601”对应笑脸的表情，在预设的字典中会被翻译为 “happy”。此外，网络用语词汇与普通文本的词汇有比较大的不同，存在类似于 LOL/LMAO/LOLROF 等有强烈情感表达的网络词汇，还包括有些被称为网络梗（gig）的特殊短语，都需要类似地添加入相应的字典转化为对应的情感表达。

3. 不合法词汇，推文中有各种链接内容，包括图片、视频、外链等等，这些链接都通过长字符串压缩技术转化为特殊短链接，作为文本夹杂在推文中。通过读取特殊的字符串(例如 “bit.ly” 为推特本身特殊的短链地址)，可以将这些字符剔除，同时 “提到” 某人这一功能会使 “@” 字符以及用户名也被加入文本，这些不合法字符也被剔除。

4. 写入 MySQL，处理之后的 Twitter 写入 MySQL 进行备份，以便需要重新计算时进行回滚。

5. 词汇划分与统计，在 Spark Streaming 中需要对句子进行划分，将句子维度的字符串转化为词汇维度的字符串，以便使用 PCA/TF-IDF 算法进行词汇统计，以及使用 word2vec 方法提供词向量输出。

6. 写入 Redis，将处理之后词汇、词向量与词汇按照 topic 分类写入不同的 Redis 实例。

上述的进一步处理流程如图 7-4 所示

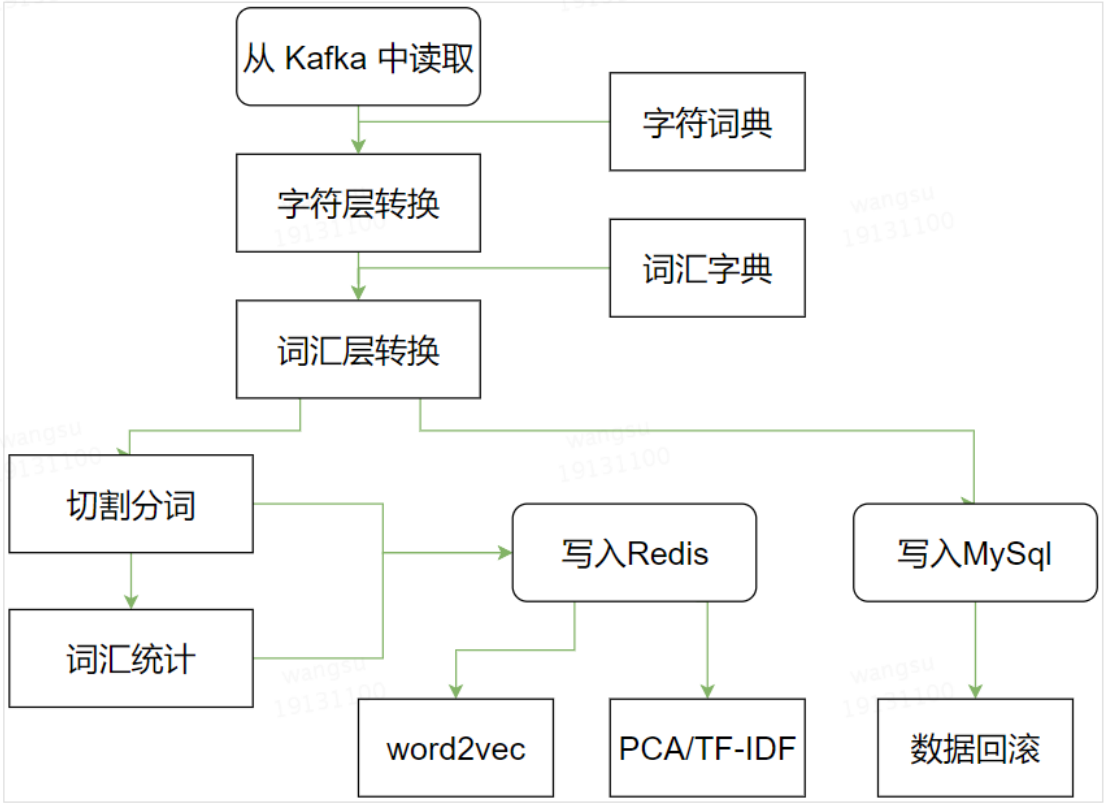

图 7-4 预处理模块数据流程示意图

Spark Streaming 支持函数式方式编程的方法，无状态无存储，仅关心转化逻辑，非常适合流式数据的处理。例如定义函数 charTransform(String[] words)，a 为待处理的字符串数组的 DStream，仅需要一行代码：

val b = a.filter(charTransform)

即可完成由 a 到处理之后的 b 的转换，并且可以通过自定的方式将函数进行级联，以获取任意处理顺序的待处理的 DStream 数据。

7.4 数据计算模块

数据计算模块的部署按照算法与主题进行区分，即计算不同的主题的实例分布在不同的物理服务器上。除此之外，不同的算法在同一个服务器中也会有不同实例：

PCA/TF-IDF 算法实例使用 python pip 库的 sklearn 模块快速实现，对于 word2vec 算法使用 python pip 库中的 gensim 快速实现。算法独立部署在相同主服务器上，而生成的词向量与词袋统计结果写入同一个 Redis 库。

PCA 算法实例需要先生成数据对应的 countVectorizer，对矩阵进行变形以方便处理，然后使用 TrancatedSVD 方法实现 PCA 算法核心内容，即指定阶数为 2，并通过酉矩阵分解的方法得到降维后的词袋数据；

TF/IDF 实例需要直接对数据进行 fit_transform 处理，然后调用 sklearn 中的 tf-idf 模块的方法可以得到降维的数据；

Word2vec 算法实例需要利用 genism 提供的事先预训练好的基于大量词向量耦合关系的模型。Gensim 模块直接提供了封装的函数，按照平均权值计算出其 embedding，按照 embedding 将词向量数据输入即可得到 SkipGram/CBOW 两种不同的模型下的 word2vec 分析结果。由于 word2vec 运算较慢，所以采用每个词向量权值相等的简化模型进行部署。

LR/Online SVM/Incremental-Decremental SVM 算法实例在模型部署的阶段，针对不同的分类器采取不同的部署策略。LR 实例较为简单，sklearn 模块中提供了封装的 LR 回归的模块。Online SVM/Incremental-Decremental SVM 实例的算法代码主要涉及到矩阵的运算，代码实现主要通过 MATLAB 工程移植到 Java 平台进行实现。实例针对算法实现了一些代码层的优化，主要优化包括：

1. 优化矩阵乘法，通过 GPU 加速的高维矩阵乘法算法 General Matrix Multiply[50]；

2. 优化增减量矩阵算法，通过多级缓存将中间变量储存起来，在中间计算种可能会被多次使用的会直接进行变形使用。

时间序列预测算法实例较为简单，实例从 Redis 中读取情感分析的中间变量，计算出给定过期时间内以及给定影响力影响下的情感评分，并获取上个时间窗口的实际股价，代入一阶线性公式，即可得到股价的预计涨跌。最后将股价涨跌预测结果写入数据库。（在结果展示阶段，由于对于股价的具体涨跌的量化预测比较困难，且误差也比较大，所以预测结果返回给前端仅返回一个可能性的涨跌的概率值。）

7.5 数据存储模块

数据存储模块的组件主要分为关系型数据库与非关系型数据库两种。

7.5.1 MySQL 关系型数据库

MySQL 数据库主要用于全量数据的持久化，批量数据的存储、读取功能，包括的库表有：

1. stock_history_price: id(主键), date(日期), code(股票代码), open(开盘价), high(最高价), low(最低价), close(收盘价), volume(交易量), ctime(创建时间)

股价数据在模型训练阶段直接载入内存参与运算，所以仅在需要重新计算

模型以及 Web 模块可视化展示时产生批量读库操作。每个交易日结束时会产生写库操作，该操作将当天股价写入数据库，由代码模块内定时任务完成。

2. stock_twitter: id(主键), twitter_id, text(推文), keyword(关键词), company_id(公司), user_id(用户 twitter_id), usser_screen_name(用户屏幕名), user_following(用户的关注数), user_follower(用户的被关注数), twitter_timestamp(时间戳), ctime(创建时间)

推文库储存了经过词汇层转换预处理之后推文的全量内容。该库数据主要为流处理系统重新计算使用。由于该库读写的并发频率都比较高，所以数据库进行了简单的读写分离。主库用于写入记录，从库用于读取记录。从库每隔一定时间会同步主库。读库的场景发生在因持久化要求需要重新计算的时候，写库场景发生在从 Kafka 读取数据进行字符层转换后。数据库主键为 id，由于按照 topic 进行读取较多，所以 company_id 设置了索引，且在维护过程中可能存在按照 twitter_id 查询的需求，twitter_id 也设置了索引。

3. stock_result：id(), company_id, prediction_result, date, ctime

结果数据主要用于存储计算完成的预测结果。

7.5.2 Redis 非关系型数据库

Redis 作为基于内存的 NoSQL 的数据库，是频繁读取与写入热点数据的解决方案。热点数据包括待计算的词向量、词袋统计结果等。由于对于多种数据结构包括 List 数据结构的良好支持，Redis 可以支持比较复杂的数据如词向量的存储。

涉及到的 Redis 数据库有：

1．twitter_text: key twitter_id, value:twitter_json.(String 类型)

JSON 格式为：

```
{   “twitter_id”:””,
    “text”: [“hello”, “world”, “stream”, “data”, “process”],
    “company_id”:”1”,
     “user_following”:”120”,
     “user_follower”:”120”,
     “twitter_timestamp”:””,
     “ctime”:”unix_time_stamp”}
```

Redis 对于内存的消耗比较大，从设计角度应当尽量节省储存空间，所以数据库仅保留了和计算直接相关的数据以及必要的维护数据。

2. twitter_sentiment: key twitter_id, value: result

Result 数据库用于存放降维与 word2vec 的计算结果。result 值为一个数组变量，在 PCA 算法中为二维数组，在其他算法可能为其他维数组。Result 数据

库的数据会被下游算法实例读取，并进行进一步的分类。

3.twitter_prediction: key: company_id, value: price_prediction

预测数据库存放时间序列的预测结果，预测结果结果会定期同步到 MySQL 数据库中提供给 Web 展示使用。

Redis 自身提供了基于 RDB 与 AOF 两种持久化方案，系统选取基于快照的 AOF 的持久化方案，确保当发生断电等物理故障时，能快速恢复备份。

除此之外，Redis 的本身的热淘汰机制会使得历史的推文数据以及历史的情感数据被清理。如果需要重新计算，可以考虑从存储预处理数据的 MySQL 数据库中读入待计算的推文重新进行计算，也可以按照中间数据的 AOF 快照多个版本进行回滚。但是 Redis 的回滚过程会造成此刻新到来的数据无法被计算，这需要结合具体情境具体考虑使用具体何种重新计算的方法。

每一个特定公司的股价预测的过程，从数据获取阶段按照不同关键词分类，到不同的公司、话题的计算模块部署在不同的机器上，最后按照话题区分了不同参数的模型，都有其特定的数据流向。但是所有的数据被统一存放在同一个 Redis 实例内，所以 Redis 在 key 值选择时必须要按照公司、话题进行区分。

整个数据计算模块的 Redis 存储方案设计如图 7-5 所示，箭头方向表示了数据的流向。

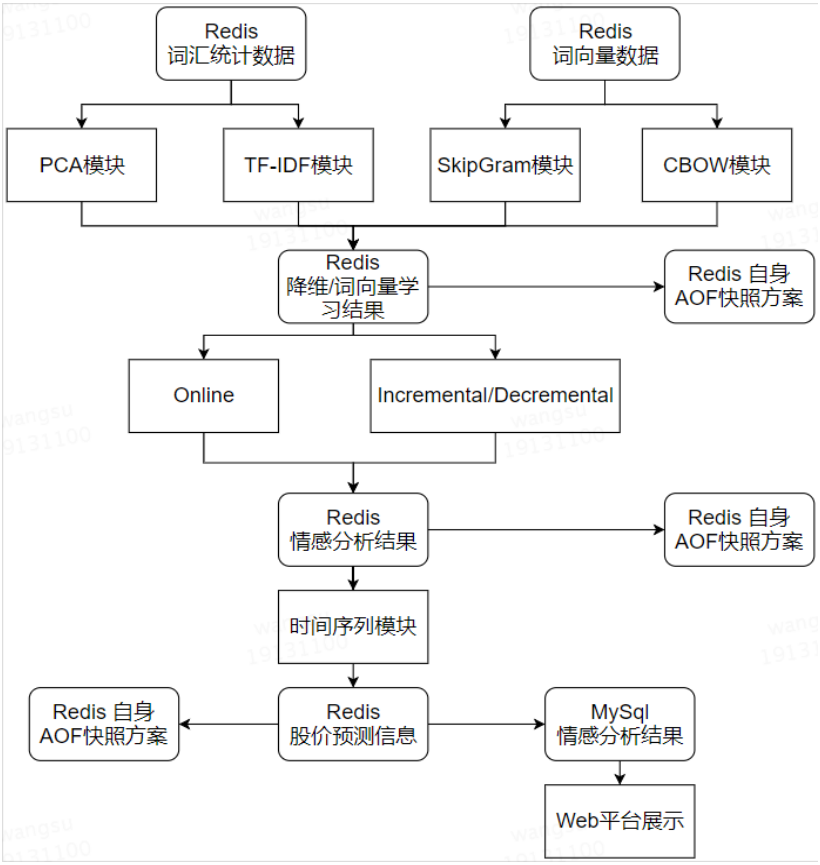

图 7-5　数据计算模块的 Redis 存储方案示意图

7.6 基于 Web 平台的大数据可视化的数据展示模块

Web 模块主要由 Spring MVC 搭建，按照 Web 项目的架构分为视图（View）、控制器（Controller）、服务（Service）三个部分：

View（即前端）：视图组件主要提供前端的展示效果，对于计算完成的可视化的结果，主要使用了 echarts 前端 js 框架，可以用比较直观的图标展示丰富数据。例如 echarts 展示股价的方案如图 7-6 所示：

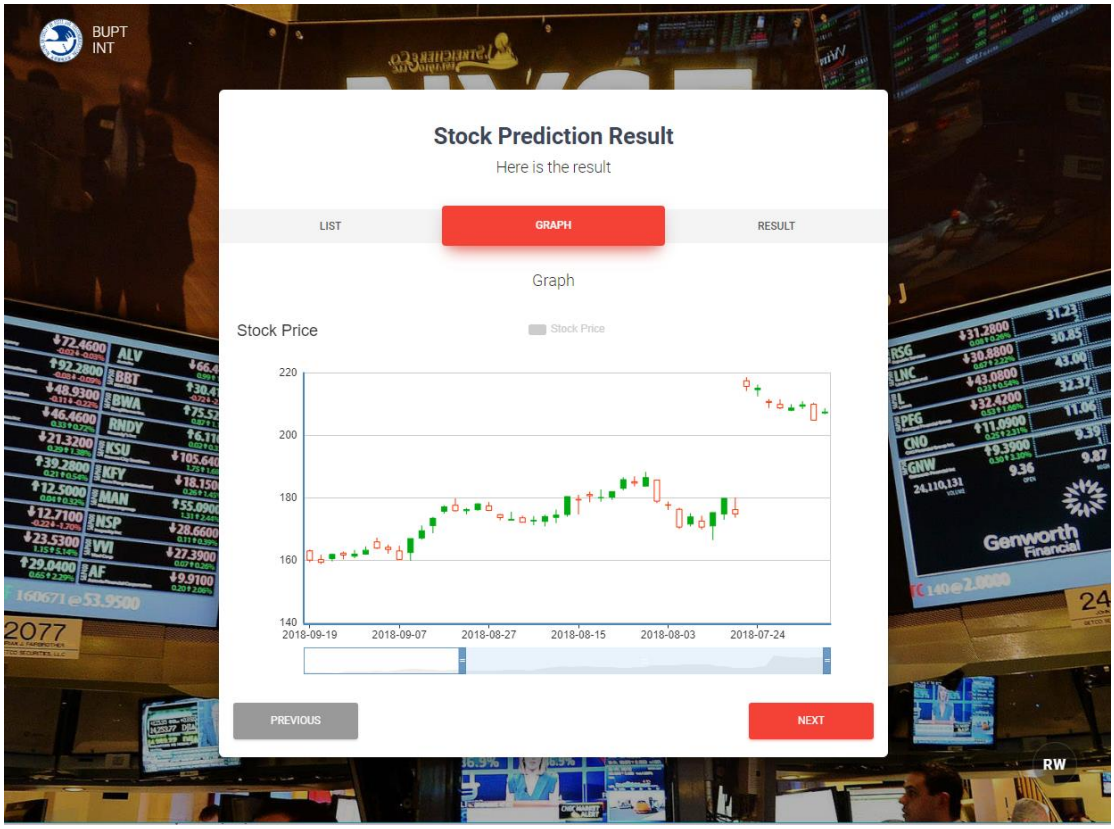

图 7-6 股价界面展示示意图

除此之外，用户与内部系统交互的设计主要在于表格的提交方式，通过三段步进式设计，用户分别选择股指、公司，数据展示时间、计算窗口长度、列表或图像展示方式，将所需要展示的形式提交给后台。其用户交互界面如图 7-7 所示：

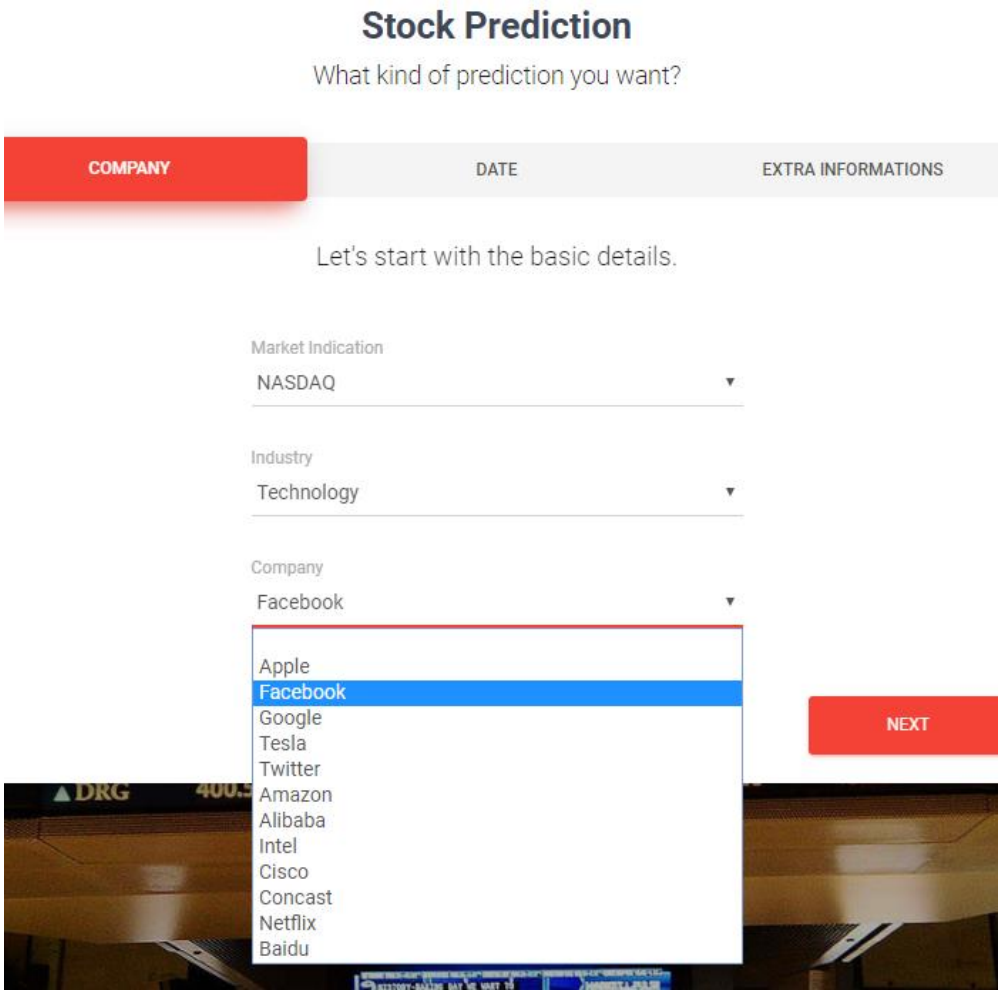

图 7-7　Web 页面表单提交示意图

Controller：控制器组件接收用户提交的表格，组装成对应的参数，并对后台服务进行调用，例如查数据库的服务，返回前端封装数据的服务，完成页面之间跳转的服务等。

Service：服务组件完成后台的业务，例如该将用户的查询请求提交给数据接入对象(Data Access Object, DAO)，并从数据库中获取对应的数据返回给 Controller。

Web 工程的业务逻辑如图 7-8 所示：

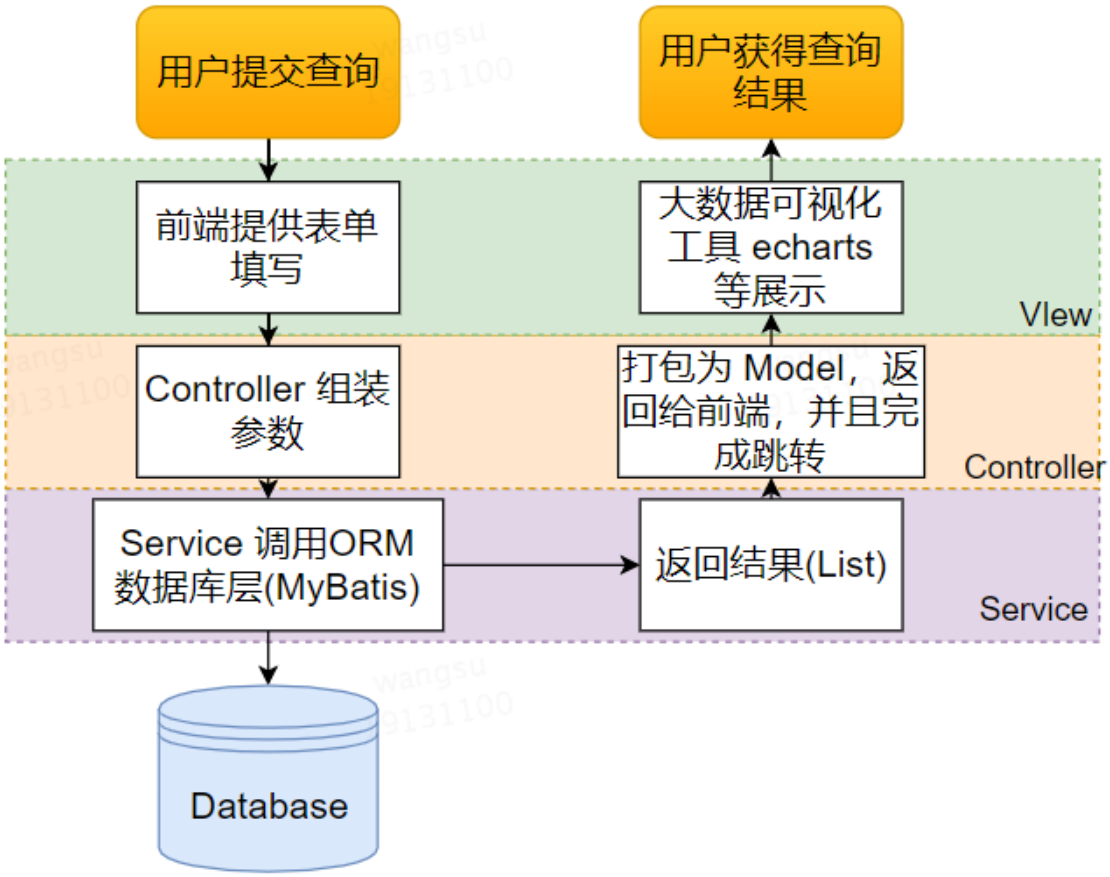

图 7-8　Web 工程架构示意图

第八章 基于流处理平台的股价预测系统的测试

本章主要对系统功能进行详细的测试，测试需要优秀完备的测试环境进行支持，测试主要包括功能性测试以及性能测试两个方面。功能测试测试系统主要模块的功能，包括单元功能测试和整体功能测试。性能测试主要包括系统的压力测试。

8.1 测试环境

本系统的测试环境都由 docker 进行装箱。docker 是一个开源的应用容器引擎，可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。使用 docker 可以方便在本地 PC 测试，然后在服务器上部署。测试的环境如表 8-1 所示：

表 8-1 测试环境

软件/框架名称	软件/框架版本	功能
Windows 10	1809	PC 操作系统
JDK	1.8	数据获取模块/Netty/Kafka/Spark Streaming 采用的语言
Python	3.6	计算平台使用的语言，大部分计算模块基于 Python.
Spring BOOT	2.0	快速搭建的 Web 框架
MySQL	5.5	关系型数据库
Redis	4.5	非关系型缓存数据库
Netty	4.1	高可用通信框架
RocketMQ	4.3.2	Kafka 消息队列
Spark Streaming	2.4	流处理框架，Hadoop 2.7

8.2 功能测试

本系统的主要功能模块需要通过单元化测试，单元化测试应该相互独立，单元测试覆盖应该尽可能全面，单元测试应该使用 Assert 断言方式对于通过有完备性的表达（通过即通过，不通过需要统一抛出携带信息的异常，而不是打印日志）。对于大部分的 Java 平台代码，采用 Mockito 与 Junit 框架实现功能测试。

8.2.1 数据获取模块

数据获取模块的测试主要测试 Tweepy 组件的调用情况，例如指定关键字能否获取推文，指定 JSON 推文能否推送到本地 Netty 的 Socket 接口，还需要考虑特殊情况下网络不可预测产生的影响。测试结果如表 8-2 所示：

表 8-2 数据获取模块主要测试结果

用例/Mock 对象	说明	结果	备注
关键字“Donald Trump”	测试根据关键字获取实时推文的功能。	符合预期	选择的关键字推文过多（如 like），会导致结果为抽样数据，数据流传输量的最大值为允许的带宽。
地区“North America”	根据用户所在地区获取热点词汇。	符合预期	和 Trends 相关的推文数量庞大，需要考虑加入过滤器筛选。
推文 JSON 报文	将获取到的 JSON 报文发到本地 Socket 用于 Netty 接收。	符合预期	缓冲区的选择至关重要，过小会导致溢出，过大会导致响应慢。
网络中断或者不可用	模拟由于代理以及 Twitter 限流等原因导致的网络不可用，应当记录日志，一段时间后重连。	符合预期	由于 Twitter 本身数据库的压力，对于历史数据读取存在流量限制，一旦触发导致 API 短时内不可用，需要根据规则设置重连时间。
网络波动	模拟由于网络原因导致 Twitter 出现波动，测试是否影响功能。	符合预期	网络的波动不可预测，对于系统的影响，可能会导致后续传输模块出现功能异常。

8.2.2 数据传输模块

数据传输模块的测试主要测试传输的功能可用，例如自定义 Netty 协议是否打包解包正常，自定义 Kafka 主题的消息是否推送成功、被消费，同时还需要考虑网络不可用的影响。测试结果如表 8-3 所示：

表 8-3 数据传输模块主要测试结果

用例/Mock 对象	说明	结果	备注
推文 JSON 报文	由 Netty 传输到本地作为 Kafka 生产者。	符合预期	该用例实际被拆分成了两个实例分别测试，然后再重新组合测试。
空报文	网络波动可能产生全空的报文。	符合预期	全空的报文不应该浪费传输带宽，应该在传输之前就被舍弃。

（续上表）

带主题的 JSON 报文	不同主题的消息由部署在分布式机器上的 Kafka 实例进行消费，也部署了不同的计算模型。	符合预期	消息应该按照主题在不同的机器上进行推送与消费。
过长的报文	有些推文经过反复转推，其报文长度远超平均的 1K 大小。	符合预期	过长的报文需要分情况特殊考虑，一般推特限制 280 个字符，但由于转推也是推文一部分，所以应当在传输时将 JSON 过长的转推字段予以删除，否则会出现缓冲区溢出的情况。
长期静默	由于上游数据获取端的网络不可用或者宕机，会导致传输端长期静默，不处理会使连接自动中断。	符合预期	一段时间没有消息传输时，Netty 双端以及 Kafka 端需要和 Zookeeper 保持心跳，以免超时连接自动断开。
较长网络时延	网络波动会导致网络出现明显时延，重发或是丢弃的策略应该作为考虑。	符合预期	网络波动可能会导致 Netty 传输时丢包，由于采用的是无责任传输机制，不需要重发。
Kafka 消息溢出	Kafka 的消费速度可能慢于消息的产生速度。	符合预期	Kafka 支持亿级消息堆积，但是由于产生的推特速度可能总是比消费速度快，所以堆积的消息可能无限制增长，这个时候需要对消息推送加上过滤器进行限速处理。

8.2.3 数据预处理模块

数据预处理模块测试主要测试字符集的转化是否成功。测试结果如表 8-4 所示：

表 8-4　数据预处理模块主要测试结果

用例/Mock 对象	说明	结果	备注
“s!1ϵncE”	字符级转换	符合预期	字符集转换需要考虑许多拉丁语系的字符可能是英文字符。
“\xF0\x9F\x98\x82”	字符级转换，laugh out with tears（笑出眼泪）	符合预期	Unicode 表示的 Emoji 表情可能是连续的十六进制字节表示，也有可能是 U+1F601 这种 64 位的模式。

（续上表）

用例/Mock 对象	说明	结果	备注
“LMAO”	词汇级转换	符合预期	需要网络语词典的支持。
“/bit.ly/2#90gh”	短链	符合预期	URL 链接的应该直接删除。
“@Kris Wu ….”	圈人功能	符合预期	没有情感倾向且可能是评论或者转发，原推较长应当直接删除。
“”	空推文	符合预期	可能因为上游处理不当空推文仍然在预处理端被接收，直接忽略。

8.2.4 数据计算模块

数据计算模块测试主要测试预处理完的推文通过部署的完毕的算法的效果。测试结果如表 8-5 所示：

表 8-5 数据计算模块主要测试结果

用例/模块/Mock 对象	说明	结果	备注
分词	对于普通推文，首先应该进行分词。	符合预期	分词结果作为中间变量应当在测试阶段展示。
词向量	对于普通推文，word2vec 应该主要考虑的自然语言转化算法。	符合预期	词向量结果作为中间变量应当在测试阶段展示。
LR/TF-IDF	完成基础功能的分类器。	符合预期	保证基础功能的分类器至少要满足在能稳定地在短时间内可以得到结果。
Online Incremental-Decremental SVM	分为淘汰策略和计算两个部分，淘汰策略维护内存中的数据池，计算需要维护模型池。	符合预期	淘汰策略需要维护监控数据，实时返回数据被消耗的速度以确保不会产生消息堆积。
Online Passive Aggressive SVM	计算阶段，计算每一步迭代结果的中间变量可以暂存一部分，起到加速作用。	符合预期	中间计算量的矩阵可以考虑在测试阶段展示。
时间序列的股价预测	维护每一次的预测结果。	符合预期	过期的预测结果需要即使淘汰。

8.2.5 数据存储模块

数据存储模块测试主要完成对于不同平台下不同 ORM 对于 DAO 对象的测试。测试结果如表 8-6 所示：

表 8-6 数据存储模块主要测试结果

用例/模块/Mock 对象	说明	结果	备注
推文数据 CRUD/MySQL	测试内容主要以增删查改为主，使用的框架为 MyBatis。	符合预期	正常
股价数据 CRUD/MySQL	测试内容主要以增删查改为主，本部分手动输入 SQL 语句导入 csv 文件。	符合预期	正常
股价预测结果 CRUD/MySQL	测试内容主要以增删查改为主，增主要通过 Redis 读后写入，两个模块联调，还需要与 Web 端进行联调。	符合预期	正常
推文数据 CRUD/Redis	测试内容主要以增删查改为主，使用的框架为 MyBatis.	符合预期	正常
推文情感结果 CRUD/Redis	测试内容以增删查改为主。	符合预期	可能涉及到一致性的问题，读取的内容并不最新，依靠 Redis 读线程的管理解决。
股价预测结果 CRUD/Redis	测试内容主要以增删查改为主，需要与 MySQL 预测结果数据库联调。	符合预期	中间计算量的矩阵可以考虑在测试阶段展示。

8.2.6 数据展示模块

数据展示模块测试主要完成 Web 平台的测试，涉及的比较简单。测试结果如表 8-7 所示：

表 8-7 数据展示模块主要测试结果

用例/模块/Mock 对象	说明	结果	备注
“http://localhost /stockquery”	主页展示	符合预期	正常
“http://localhost /stockquery&…”	提交表单展示	符合预期	正常

8.3 非功能测试

如前文所阐述，流处理系统需要解决的问题包括：实时性以及高并发的问题，所以性能评估的内容主要从这两个方向进行展开：

8.3.1 时延测试

时延测试在系统正常负载（QPS=10K）情况下，测试了多个主要模块的处理时延。如表 8-8 所示：

表 8-8 系统主要模块时延测试结果

时间窗口	时延种类	时延	备注
90min	平均获取并处理时延	167 ms	处理时延来自于接口流量前后时间差。
	平均传输时延	416 ms	
	平均处理时延（预处理+写库）	12ms	
	平均计算时延	1472ms	
45min	平均获取并处理时延	154ms	传输时延来自于网络中传输的时延。
	平均传输时延	298ms	
	平均处理时延（预处理+写库）	18ms	
	平均计算时延	1902ms	
15min	平均获取并处理时延	177ms	计算时延为计算平台得出计算结果的时间。
	平均传输时延	144ms	
	平均处理时延（预处理+写库）	39ms	
	平均计算时延	1678ms	

可以看到，数据前期准备工作时延主要来自于传输时延，整个系统的时延主要来自于计算时延。计算时延为每一个时间窗口计算时间除以分批数据的批数。

8.3.2 压力测试

在进行压力测试之前需要确定测试的目标。为了满足系统高并发以及高可用的的特点，测试的目标为：

1. 找到流处理系统能处理的最大的数据带宽，而且功能可用；
2. 找到流处理系统高负载运行状态下，数据带宽与时延的关系；
3. 找到流处理系统超 50%设计负载运行状态下，两次宕机之间的时间间隔；
4. 找到大流处理系统高负载运行状态下，造成数据瓶颈的模块。

针对以上四个目标，分别设计了评估。

8.3.2.1 最大数据带宽

通过模拟 HTTP Post 推文输出到数据获取端进行数据带宽的压力测试，同时保证整体时延在 30000 ms 内，得到 3 次平均的最大数据带宽以及 QPS 如表 8-9 所示：

表 8-9 系统最大数据带宽以及 QPS 测试结果

计次	数据获取带宽（1KB=1 条）	Kafka 消费速度（条）	计算速度（条）
1	22.2MB/s	17.4K/s	17.4K/s
2	25.1MB/s	19.7K/s	19.7K/s
3	19.9MB/s	16.2K/s	16.2K/s

8.3.2.2 数据带宽与时延

通过模拟 HTTP Post 推文不断从正常负载增加到最大负载，输出到数据获取端，可以得到数据带宽与时延的压力测试曲线大致图 8-1 所示：

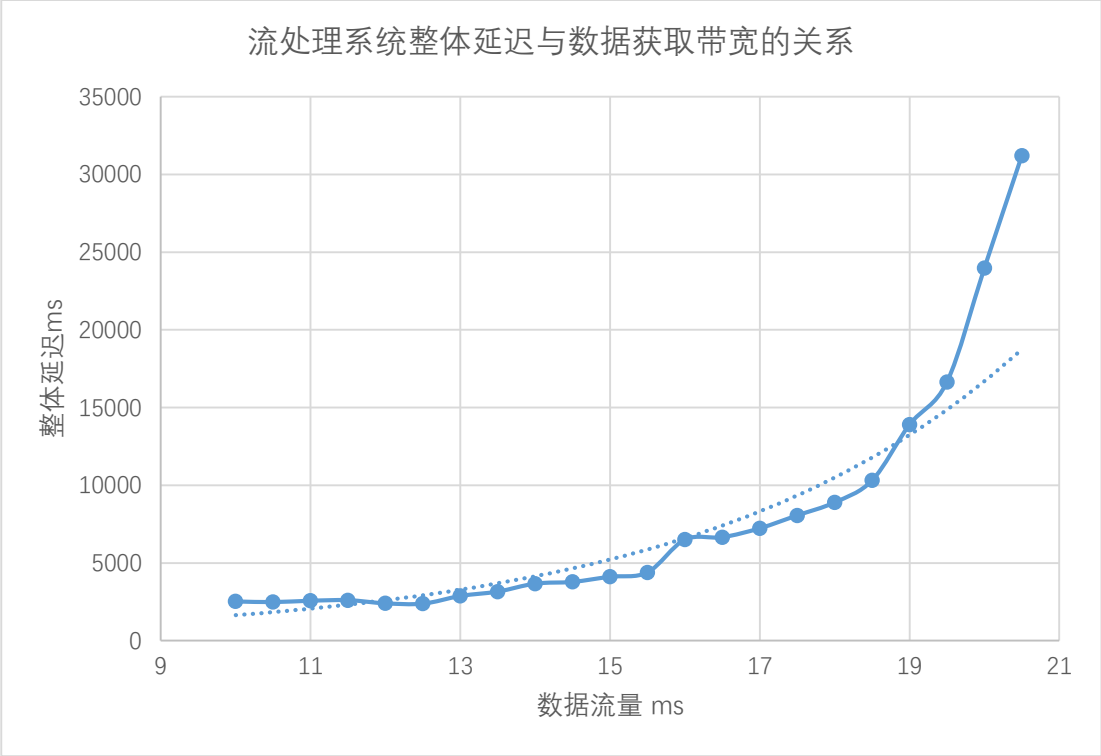

图 8-1 流处理系统整体延迟与数据获取带宽的关系

根据实践认为一般认为系统时延超过 5000ms 时系统稳定性与可用性会受到比较大的影响。

8.3.2.3 宕机时间间隔与瓶颈

根据 8.3.2 的压力测试，可以看到数据流量超 50%时，整体时延会超过 5000ms，消息可能出现堆积现象，网络通信可能会出现故障现象，传输的模块会开始大量丢弃发送报文，如果来不及丢弃可能会出现消息队列溢出。于是将系统以超出设计负载 50%流量持续运行，观察模块响应。宕机时间间隔大约在 3h-12h 左右。同时发现以下几个模块会出现不可用的问题：

1. Kafka 消息堆积报警；
2. Netty 最大线程池报警；
3. 计算平台出现计算死循环，反复重启，导致无法消费。

上述的问题即为在性能上可能造成的瓶颈。所以针对以上的三个瓶颈，分

别进行了不同优化。

针对 RocketMQ，消息报警消息提高阈值，提前通知生产者降低生产速率。

针对 Netty 最大线程，修改 Linux Ulimit 并且启用服务器多核优化

针对计算出现的死循环，增加丢弃策略，使用快慢指针记录计算节点，出现死循环立即丢弃当前运算数据。

第九章 结束语

本章主要对本论文进行归纳性总结，包括论文工作中在算法以及工程上的实现。同时针对工作中不足的地方，指出未来工作可以提升以及修改的地方。

9.1 工作总结

通过深入研究股票预测与社交网络情感的相关问题，本文发现现有的基于社交网络情感分析的，且针对于特定公司股价实时预测的相关研究比较少，而且其研究意义比较重大。于是本文从实时社交网络情感分析出发，提出、设计并实现了一套能对特定公司实现股价实时预测的流处理系统。

本文在算法上提出了改进数据更新淘汰策略的 Incremental/Decremental SVM 算法与最小化更新步长的 Online Passive Aggressive SVM 算法。上述算法的关键是确定历史数据以及新到来的数据如何“分担”影响模型的比例。两种上述的算法均可以通过调整某些参数条件，来实时动态更新新到来的数据以及历史数据的对模型影响比例，使得算法能够充分利用时刻变化的社交网络的舆论情感数据进行股价预测。本文还重新设计了基于时间序列的股价预测算法，将情感分析结果作为参量从而引入股价预测的结果。

本文对提出的算法进行验证和评估，发现算法的在线计算的特性能够较好的适应实时的社交网络情感数据，其运行时延与模型的适应能力等指标优于其他的算法，其准确率达到系统的设计要求。

本文基于上述提出的算法搭建了一套股价实时预测系统，主要解决了数据获取，数据传输，数据处理以及数据展示等问题，并对预测系统进行了比较完备的功能测试与压力测试，达到了系统的高可用性与低时延性等预期效果。

9.2 未来工作

本论文工作仍有一些尚待进一步解决的问题，可在未来的工作中解决。

1. 寻求更完善的数据源。在研究实践工作过程中，Twitter 网站曾多次变更 API，多次修改最大允许爬取的数据量，此情况直接导致数据获取端的接口多次修改，所以部分数据在时间上存在不连续的情况。在未来可以考虑增加数据的获取渠道，如 RSS 订阅的新闻客户端以及新闻评论，Google 开放的实时搜索热度 API 等多个数据源。

2. 提出更具代表性的情感分析算法以及股价预测算法。本文工作针对在纳斯达克证券交易所以及纽约证券交易所上百家公司进行了跟踪，有大约 30%-40% 的公司会有比较好的股价预测结果。剩余的 60%-70%的公司的股价和舆论的关系没有预计中的密切。由于平台的运算资源受限，大量运算资源提供给在线算法，在未来可以考虑同时在计算平台上部署更多的情感分析算法实例，以便覆盖对更

多的上市公司的预测。

参考文献

[1] Kotler, P., & Levy, S. J. (1969). Broadening the concept of marketing. [J]. Journal of marketing, 33(1), 10-15.
[2] Fama, E. F. (1995). Random walks in stock market prices. [J]. Financial analysts journal, 51(1), 75-80.
[3] Shiller, R. J. (2015). Irrational exuberance: [M]. Revised and expanded third edition. Princeton university press.
[4] https://en.wikipedia.org/wiki/Subprime_mortgage_crisis
[5] Kietzmann, J. H., Hermkens, K., McCarthy, I. P., & Silvestre, B. S. (2011). Social media? Get serious! [J]. Understanding the functional building blocks of social media. Business horizons, 54(3), 241-251.
[6] Zhang, Z., Shen, Y., Zhang, G., Song, Y., & Zhu, Y. (2017, November). Short-term prediction for opening price of stock market based on self-adapting variant PSO-Elman neural network. [A]. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS) [C]. 225-228.
[7] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. [J]. Journal of computational science, 2(1), 1-8.
[8] https://en.wikipedia.org/wiki/United_Express_Flight_3411_incident
[9] Edwards, R. D., Magee, J., & Bassetti, W. C. (2018). Technical analysis of stock trends [M]. CRC press.
[10] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. [J]. Journal of computational science, 2(1), 1-8.
[11] Sul, H., Dennis, A. R., & Yuan, L. I. (2014, January). Trading on twitter: The financial information content of emotion in social media. [A]. In 2014 47th Hawaii International Conference on System Sciences [C]. 806-815.
[12] http://www.internetlivestats.com/twitter-statistics/
[13] Wallach, H. M. (2006, June). Topic modeling: beyond bag-of-words. [A] In Proceedings of the 23rd international conference on Machine learning [C]. 7-984. ACM.
[14] https://en.wikipedia.org/wiki/Principal_component_analysis
[15] https://en.wikipedia.org/wiki/Tf%E2%80%93idf
[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. [J]. In Advances in neural information processing systems 3111-3119.
[17] Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. In Advances in neural information processing systems [J]. 409-415.
[18] Das, S. (1994). Time series analysis. [M]. Princeton University Press, Princeton, NJ.
[19] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992).

Class-based n-gram models of natural language. [A]. Computational linguistics, [C]. 18(4), 467-479.
[20] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. [A] In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). [C]. 1532-1543.
[21] Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. [J] Neural processing letters, 9(3), 293-300.
[22] Amihud, Y. (2002). Illiquidity and stock returns: cross-section and time-series effects. [J] Journal of financial markets, 5(1), 31-56.
[23] https://netty.io/
[24] https://kafka.apache.org/
[26] https://spark.apache.org/docs/latest/streaming-programming-guide.html
[27] https://redis.io/
[28] https://spring.io/projects/spring-boot
[29]亓开元,赵卓峰,房俊,马强.针对高速数据流的大规模数据实时处理方法. [J]. 计算机学报,2012,35(03):477-490.
[30] Tang, D., Wei, F., Qin, B., Zhou, M., & Liu, T. (2014). Building large-scale twitter-specific sentiment lexicon: A representation learning approach. [A] In Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers. [C] 172-182.
[31] Barsade, S. G. (2002). The ripple effect: Emotional contagion and its influence on group behavior. [J] Administrative science quarterly, 47(4), 644-675.
[32] Le Bon, G. (1900). Psychologie des foules. [M] F. Alcan.
[33] Tang, D., Wei, F., Qin, B., Zhou, M., & Liu, T. (2014). Building large-scale twitter-specific sentiment lexicon: A representation learning approach. [A] In Proceedings of coling 2014, the 25th international conference on computational linguistics: Technical papers. [C] 172-182.
[34] Linderman, S., & Adams, R. (2014, January). Discovering latent network structure in point process data. [A] In International Conference on Machine Learning. [C] 1413-1421.
[35] 周伟军. 拟牛顿法及其收敛性. [D]. 湖南大学,2006.
[36] White, H. (1982). Maximum likelihood estimation of misspecified models.[J] Econometrica: Journal of the Econometric Society, 1-25.
[37] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. [J] Journal of Machine Learning Research, 7(Mar), 551-585.
[38] https://data.world/crowdflower/emotions-about-nuclear-energy
[39] https://data.world/crowdflower/brands-and-product-emotion
[40] https://data.world/crowdflower/apple-twitter-sentiment
[41] https://data.world/crowdflower/airline-twitter-sentiment
[42] https://data.world/datafiniti/consumer-reviews-of-amazon-products
[43] https://data.world/crowdflower/weather-sentiment

[44] https://data.world/crowdflower/sentiment-analysis-in-text
[45] https://www.nasdaq.com
[46] https://www.nyse.com
[47] https://www.juhe.cn
[48]Reilly, Frank K., and Keith C. Brown. Investment analysis and portfolio management. [M] 中信出版社, 2002.
[49] http://dev.twitter.com/apps
[50] https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms#Level_3

致谢

攻读硕士学位期间发表和录用的论文