计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(7):264−267 [doi: 10.15888/j.cnki.csa.007519]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 系统日志故障预测中的 ELK 与 LSTM 应用与实践①

徐志斌[1], 叶　晗[1], 王　晗[2], 郜义浩[2]

[1](北京市首都公路发展集团有限公司, 北京 100161)
[2](北京云星宇交通科技股份有限公司, 北京 100078)
通讯作者: 郜义浩, E-mail: gaoyihao@bytd.cn

**摘　要**: 随着业务系统规模不断扩大, 系统结构也变得十分复杂, 常规基于规则的方法已经很难判断多个系统相互作用下的复合型故障, 也难以对潜在故障进行预测. 本文在多业务系统的复杂场景下, 使用 ELK 平台对日志进行集中化管理, 梳理出复杂系统环境下日志与各业务系统、主机、进程之间的关系, 筛选出系统中直接与故障相关的日志文件, 进而在深度学习框架 TensorFlow 中使用这些海量数据对 LSTM 算法模型进行训练, 从而实现对系统的实时故障预测.

**关键词**: ELK; LSTM; 故障预测; 深度学习; TensorFlow

引用格式: 徐志斌,叶晗,王晗,郜义浩.系统日志故障预测中的 ELK 与 LSTM 应用与实践.计算机系统应用,2020,29(7):264–267. http://www.c-s-a.org.cn/1003-3254/7519.html

## Application and Practice of ELK and LSTM in System Log Fault Prediction

XU Zhi-Bin[1], YE Han[1], WANG Han[2], GAO Yi-Hao[2]

[1](Beijing Capital Highway Development Group Co. Ltd., Beijing 100161, China)
[2](Beijing Yunxingyu Traffic Technology Co. Ltd., Beijing 100078, China)

**Abstract**: As the scale of systems continues to expand, the system structure also becomes very complex. The rule-based methods have been difficult to judge the composite faults under the interaction of multiple systems, and it is also hard to predict potential faults. Firstly, the study uses the ELK platform for centralized management of logs in complex scenarios of multi-business systems. Then, it sorts out the relationship between logs and various business systems, hosts, and processes in a complex system environment. Finally, we filter out the log files related to the failure in the system, and use these data in the deep learning framework TensorFlow to train the LSTM algorithm model, so as to realize the real-time fault prediction of the system.

**Key words**: ELK; LSTM; failure prediction; deep learning; TensorFlow

## 1　引言

随着现实业务规模的扩大, 对应的信息系统也变得越来越复杂. 另一方面, 业务人员对于系统稳定性的要求却越来越高, 这对运维的质量提出了很高的挑战. 一旦业务系统发生故障, 运维人员很难精准定位故障发生的原和快速修复系统. 传统监控系统通常难以发现潜在的故障, 因为在业务系统非常复杂的情况下, 每一时刻都会产生大量有关或无关的运行状态数据、日志数据, 使得通过规则编码或者人工排查进行故障预警变得越来越不显示.

本文通过两个步骤完成基于日志的系统故障预警工作的研究. 包括:

① 收稿时间: 2019-12-05; 修改时间: 2020-01-14, 2020-01-21; 采用时间: 2020-02-11; csa 在线出版时间: 2020-07-03

(1) 借助 ELK 平台收集汇总各个系统中的日志, 实现日志的集中化管理. 这样一方面可以方便运维人员的日常查询、搜索、管理日志, 提高运维效率. 另一方面也可提供后续研究打下数据基础, 为日志预警模型筛选出确实有效的原始数据.

(2) 基于 LSTM 模型, 通过对原始数据的清洗、规范化等工作, 结合历史环境下系统相关告警信息, 进而训练得到基于日志的故障预警模型.

研究内容对于发现潜在故障, 快速定位故障来源, 准确识别故障原因, 缩短故障恢复时间, 提高运维人员工作效率都具有一定的帮助.

## 2 研究现状

### 2.1 日志分析工具

传统上通常直接使用 Shell 或者 Python 脚本对单机日志进行分析. 当这种方式不能满足海量日志分析要求的时候, 分布式的日志采集、分析系统应运而生, 并且逐渐成为了各大互联网企业的标准配置[1].

当前主流的日志管理系统通常由采集端和服务端两部分构成.采集端通常为在各主机安装的代理 (agent) 或客户端应用 (client), 这些代理或客户端将采集到的日志数据推送到服务端. 服务端则实现方式众多, 有的单纯提供存储、查询功能, 也有提供存储、计算分析、查询等复杂功能的庞大的分布式系统[2,3].

常见的日志管理系统有 Cloudera 开源的 Flume[4], Elastic[5]公司开源的 ELK Facebook 开源的 Scribe[6] 和 Apache 的 Chukwa[7]等. 本文选用 Elastic 公司开源的 ELK 套件来实现日志的采集、查询和管理.

### 2.2 深度学习及框架

深度学习 (deep learning) 是机器学习的分支, 是一种以人工神经网络为架构, 对数据进行表征学习的算法. 其最大的好处是使用无监督或半监督的特征学习和分层特征提取算法来替代手工获取特征, 从而大大减少研究人员对研究领域知识的依赖.

当前深度学习中有很多成功的模型, 例如深度神经网络、卷积神经网络、循环神经网络等等. 这些模型广泛应用于多个研究领域, 在计算机视觉、语音识别、自然语言处理以及生物信息学等领域都取得了极好的效果.

针对各种成熟的深度学习模型, 研究者们提出了多种不同的框架, 目前使用最广泛的包括 TensorFlow、Caffe、Keras、PyTorch 等. 其中 TensorFlow 由 Google 公司出品, TensorFlow 框架更新维护频繁, 同时具备 python 与 C++接口, 同时具备完整教程, 收到了广大开发工作者和学术人员的青睐, 目前已经成为深度学习框架中最受欢迎的选择. 本文所有模型训练工作均采用 TensorFlow 框架完成.

## 3 模型选型与模型构建

### 3.1 模型选型

(1) CNN

CNN 开始普遍用于图像识别中, Yoon Kim 在 2014 年提出了 TextCNN, 将卷积神经网络 CNN 应用于文本分类中, 利用多个不同 size 的 Kernel 来提取句子中的关键信息. 但 CNN 有个最大问题是固定 filter_size 的视野, filter_size 的超参调节很繁琐. 另外由于 CNN 的特性, TextCNN 对词的顺序很不敏感, 无法从上下文中获取信息.

(2) RNN

RNN 模型结构最大的特点在于会将上一时刻的结果作为当前时刻的输入, 基本模型结构可以用以下公式表示:

$$h_t = \tanh\left(W\begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}\right) \tag{1}$$

其中, $h_t$为当前时刻输出, $h_{t-1}$为上一时刻输出, $x_t$为当前时刻输入, $W$即为需要训练的权重[8].

RNN 相比于 CNN 在文本上最大的优势在于考虑到了上文的信息. 从公式中可以看出, $t$时刻的输出与 $1,2,\cdots,t-1$全部有关, 但这也同时导致$h_t$是由一系列 $W$连乘得到的, 进而引发梯度消失和梯度爆炸的问题. 因此, 当文本的篇幅比较长的时候, RNN 的缺点就会凸显出来. 日志信息往往携带者一连串的报错文本, RNN 难以满足当前要求[9].

(3) LSTM

LSTM (Long Short-Term Memory) 可以认为是 RNN 的一种特殊形式, 模型结构如图 1 所示.

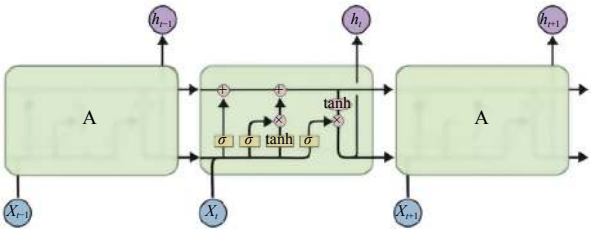


图 1 LSTM 网络模型结构

LSTM 模型通过门控状态来对信息进行选择性的记忆, 满足了需要长时间记忆信息和遗忘信息的需求.

LSTM 弥补了 RNN 在长文本中存在的缺点, 即可以选择性遗忘不重要的信息, 在长文本中只保留重要信息的关联性. 本文基于日志的故障预测模型就是以 LSTM 为基础完成的.

### 3.2 模型构建

深度学习模型构建可以简化为 4 部分内容, 即模型结构的选择、激活函数的选择、损失函数的选择以及下降梯度的选择. 本文在模型上使用标准的 LSTM 模型, 在模型结构上不需要进行额外设定.

模型在损失函数上采用 Softmax, 即交叉熵损失, 相比于范数损失、均方误差等, 交叉熵损失在分类上收敛速度要更加迅速.

模型在梯度下降上采用 SGD 的方式. 需要提及的是这里分别使用过 Adam 算法、AdaDelta 算法和几种 GD 方法. Adam 与 AdaDelta 方法虽然收敛速度更快, 但是由于训练数据本身的质量一般, 最终模型的准确率往往很差, 而 GD 的收敛速度虽然最慢, 但最终得到的效果要优于其他几项, 为了避免 GD 出现局部最优的现象, 最终选定 SGD 方法.

## 4 系统案例分析

本文选取首发集某业务系统作为试点, 实现系统基于日志的故障预测完整流程, 包括日志分析、日志处理、模型训练与验证工作.

### 4.1 日志文件分析

该业务系统的不同日志文件中, 虽然告警内容的格式也有所不同, 但告警信息内容的结构基本上是相同的.

如图 2 所示, 所有告警日志基本包含以下几部分内容.

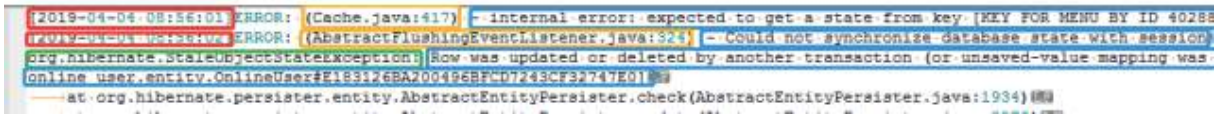


图 2 日志文件信息

(1) ERROR_time: 告警日志产生的时间

(2) ERROR_label: 告警标签, 显示当前错误的类别, 这个类别信息是日志自己生成的, 而非通过告警产生的人工标签.

(3) ERROR_label_detection: 部分告警会有简单的描述或解释, 这一部分信息不是必须的.

(4) ERROR_content: 告警内容本身, 这是用于模型训练的关键数据.

### 4.2 模型训练

(1) 数据准备

本文总共选取了 18 年 1 整年的 tomcat 告警日志作为训练数据, 日志每天产生告警信息 20~50 条不等, 共计训练数据 8000 多条. 同时通过采集到的告警信息对日志数据进行标定, 将数据分为产生预警和不产生预警两类.

(2) 数据处理

对日志数据进行处理, 包括结构化、时间格式标准化、日志格式转化、文本序列化、文本长度截取等工作, 最终得到可以直接进入模型的训练数据.

(3) 参数调整

模型主要需要调整的参数包括隐层数量、遗忘门偏置 (drop 值)、最大序列化值 (embedding_size) 以及最大文本长度.

隐层数量决定模型可以识别内容的复杂程度上限, 层数越多可以完成的分类越复杂, 但是耗费的时间与资源也会相应增加, 经过调整, 隐层数量为 40 时已经可以满足模型预测目标.

遗忘门偏置一般要求大于等于 1.0 即可, 当模型容易陷入局部最优时可以适当增大, 本文在构建模型时已经选择了不易出现局部最优的 SGD 梯度, 因此设置为 1.0 即可.

embedding_size 的合理大小是需要在模型训练过程中不断去调整的, 值过大会导致加入很多干扰项, 影响模型准确率; 值过小则进入训练的数据信息太小, 同样会造成模型准确率下降. 经过反复调试, 故障预警模型的 embedding_size 在 800~1000 之间准确率最高, 目前模型的 embedding_size 值为 1000.

最大文本长度同样会影响最终训练模型的质量, 一般情况下的最大文本长度可以设置为最长文本的 60% 到 70% 左右, 但对于日志文件来说不同文件间的长度差异过大, 需要反复多次试验才能得到最佳值, 本次故障预警模型使用的最大文本长度为 500.

对于不同模型也需要对上述参数反复调整, 最终保证训练模型达到最优.

(3) 模型训练

模型开始训练后会输出运算步数以及 loss 值, 作为准确率参考依据, 如图 3 所示.

```
step = 3500      mean loss = 0.6526153683662415
step = 3600      mean loss = 0.6536033153533936
step = 3700      mean loss = 0.6584043502807617
step = 3800      mean loss = 0.6576941609382629
step = 3900      mean loss = 0.6491097211837769
step = 4000      mean loss = 0.6579610705375671
step = 4100      mean loss = 0.6501691341400146
step = 4200      mean loss = 0.649808943271637
step = 4300      mean loss = 0.6514708995819092
step = 4400      mean loss = 0.6518977880477905
step = 4500      mean loss = 0.6513704061508179
step = 4600      mean loss = 0.6558107137680054
step = 4700      mean loss = 0.6590390801429749
step = 4800      mean loss = 0.6564667224884033
step = 4900      mean loss = 0.6572102904319763
step = 5000      mean loss = 0.6513528823852539
step = 5100      mean loss = 0.6512566804885864
```

图 3 训练过程样例

当 step 增加到一定程度后, 模型的 loss 值就不再下降了, 一般情况下需要经过 3W–5W 次的迭代. 模型训练最终生成文件样例如图 4.

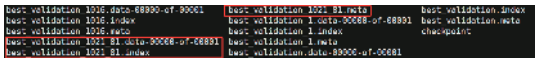

图 4 训练模型结果展示

### 4.3 结果验证

模型训练完成后随机选取 4096 条未进行训练的数据进行故障预警模型测试.

测试结果如图 5 所示.

```
serious => serious
null => null
serious => serious
serious => null
null => null
serious => serious
serious => serious
serious => serious
serious => serious
serious => serious
serious => null
serious => serious
null => null
null => null
serious => null
null => null
null => null
null => null
3376 4096 0.82421875
omnisky@omnisky:/home/wanghan/log_warn$
```

图 5 故障预测模型测试集准确率

图 5 中左侧表示日志实际分类状态, 右侧表示日志预测分类状态, 测试数据共计 4096 条, 其中识别正确的数据为 3376 条, 故障预测模型最终准确率为 82%.

## 5 结论

本文首先采用 ELK 工具实现了日志采集、存储工作, 为后续故障识别与模型训练提供基础. 之后, 使用这些日志在 TensorFlow 框架中对标准 LSTM 模型进行训练, 最终实现故障识别模型. 经测试, 障预测模型最终准确率为 82%, 准确性良好.

本文采用的整套流程具备很强的可操作性, 可用于实际系统相关的故障预测工作, 可为运维人员提供工作效率、保障业务系统稳定性提供一定帮助. 同时也存在一些不足, 例如尚未考虑日志与日志、告警与告警之间的关联性问题, 模型训练中未对词汇表的进一步研究与调整工作等, 这些都可以作为后续的研究内容, 帮助进一步提升模型的准确率.

## 参考文献


1 陈飞, 艾中良. 基于 Flume 的分布式日志采集分析系统设计与实现. 软件, 2016, 37(12): 82–88. [doi: 10.3969/j.issn.1003-6970.2016.12.019]

2 Goetz PT, O'Neill B. Storm 分布式实时计算模式. 董昭, 译. 北京: 机械工业出版社, 2015. 1–9.

3 陆世鹏. 基于 Spark Streaming 的海量日志实时处理系统的设计. 电子产品可靠性与环境试验, 2017, 35(5): 71–76. [doi: 10.3969/j.issn.1672-5468.2017.05.014]

4 拉斐尔, 酷奇. 深入理解 Elasticsearch. 张世武, 译. 北京: 机器工业出版社, 2017. 18–11.

5 Scribe. https://github.com/facebookarchive/scribe/wiki. [2018-09-17].

6 张川, 邓珍荣, 邓星, 等. 基于 Chukwa 的大规模日志智能监测收集方法. 计算机工程与设计, 2014, 35(9): 3263–3269. [doi: 10.3969/j.issn.1000-7024.2014.09.055]

7 刘季函. 基于 Spark 的网络日志分析系统的设计与实现[硕士学位论文]. 南京: 南京大学, 2014.

8 Socher R, Lin CCY, Ng AY, *et al*. Parsing natural scenes and natural language with recursive neural networks. Proceedings of the 28th International Conference on Machine Learning(ICML11). Bellevue, WA, USA. 2011. 129–136.

9 İrsoy O, Cardie C. Deep recursive neural networks for compositionality in language. Proceedings of the 27th International Conference on Neural Information Processing Systems. Cambridge, MA, USA. 2014. 2096–2104.