基于 NLP 的兴趣点数据上线系统设计与实现

张先荣 郑贵俊
(中国科学技术大学软件学院 安徽 合肥 230051)

摘 要 全面丰富的兴趣点(Point of Interest,POI)数据直接影响着地图 App 厂商的地理位置服务。针对传统的 POI 数据采集与上线方式周期长、速度慢的问题,提出一种高效的采集、上线 POI 数据的方式。将数据上线工作细化为:数据采集,数据格式化,数据判重与存储。在数据采集模块上采用一种负载均衡的分布式网络爬虫采集技术,数据格式化模块用于处理数据采集模块采集出的原始数据格式不统一的问题。数据判重模块将新旧数据的名称进行相似度计算,再结合经纬度计算的距离进行判重。结合 Word2Vec 与 Siamese-LSTM 设计判重模型,准确率达 93.5%。

关键词 数据采集 数据判重 POI 数据 Word2Vec Siamese-LSTM 短文本相似度

中图分类号 TP3 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.12.004

DESIGN AND IMPLEMENTATION OF POI DATA ONLINE SYSTEM BASED ON NLP

Zhang Xianrong Zheng Guijun
(*School of Software Engineering, University of Science and Technology of China, Hefei 230051, Anhui, China*)

Abstract The comprehensive and abundant POI (Point of Interest) data directly affects the geographical location services of map App manufacturers. Aiming at the problems of long cycle and slow speed of traditional POI data collection and upload mode, an efficient way of collecting and upload POI data is proposed. The data upload work was divided into data collection, data formatting, data uniqueness and storage. The data collection module adopted a load balanced distributed Web crawler collection technology, and the data formatting module was used to deal with the inconsistency of the original data format collected by the data collection module. The data uniqueness module calculated the similarity between the old and new data names, and then judged the uniqueness by combining the distance calculated by longitude and latitude. Combining Word2Vec with Siamese-LSTM to design the uniqueness model, the accuracy is 93.5%.

Keywords Data collection Data uniqueness POI data Word2Vec Siamese-LSTM Short text similarity

0 引 言

地图位置 POI 数据在电子地图中扮演着重要角色。常见的电子地图中存在两种数据,一种是底图数据,另一种是地图位置数据——POI 数据。底图数据一般在电子地图中描述大的轮廓,如哪一片是小区,哪一片是公园。底图数据一般在国家地理测绘局购买,这些数据不需要人为测量,由卫星照片合成。

一个电子地图的好坏一般是由 POI 数据决定。POI 数据严格意义上属于向量数据,该数据与底图数据最大的不同是不需要描述建筑物或者实体的轮廓,在地图上呈现的是一个点的标注,这些点都是坐标点。如银行、餐厅和咖啡馆等,这些都可以用 POI 坐标表示,一般比较容易更新或者编辑。如果用面向对象的思维考虑 POI 数据,其属性应该包括:数据编号(Id),数据名称(Name),经度(Longitude),纬度(Latitude),国别(Country),省(Province),市(City),县区(County)

收稿日期:2019-07-16。**张先荣**,硕士生,主研领域:机器学习,软件工程。**郑贵俊**,硕士生。

和地址(Address)等。

因此,本文提出并设计一种 POI 数据采集与判重系统,利用程序实现自动数据采集,能够满足地图厂商实时更新的需要,尤其为企业节约了大量的成本,具有重要的工程设计价值意义。

1 研究现状

1.1 网络爬虫

本文系统的核心是数据采集与判重,数据采集技术主要是主题网络爬虫[1-2]。相对于全网爬虫,该技术更准确高效,目前主要有以下几类:

(1) 基于内容的启发式方式。Best first search 算法[3]:将需要采集的网页的 URL 加入到一个优先级队列中,队列的优先级是根据主题和网页内容的相关性计算的。Cho 等[4]把等待采集的 URL 队列又细分成两个队列,一个队列叫作 hot_queue,另一个叫作 URL_queue。仅当 hot_queue 相关网页的数据采集结束之后,才开始采集 URL_queue 队列。算法缺点是当主题词比较多时,效率特别低。Fish search 算法[5]:将待采集的网页看成一群鱼。如果有相关主题信息,鱼就有"食物",鱼有了"食物"就会"繁殖"。没有相关主题信息,鱼则会出现死亡。算法的缺点是相关度计算设置成离散,计算不够准确,相关度相同的网页容易被忽略掉。Shark search 算法[5]继承了 Fish search 方法,相关度度量方法不是利用离散值,而是利用 0 ~ 1 之间的相关度。在计算相关性时,还充分利用了锚文字。这样能更好地提高爬行方向的正确性。

(2) 基于超链图方式。Page Rank 算法[7]利用一个网页的重要性传递到了它所引用的网页,因此它被称为权威网页。

$$R(i) = (1-d) + d \times \sum_{j \in B(i)} \frac{R(j)}{N(j)} \tag{1}$$

式中:$B(i)$ 表示那些指向网页 i 的网页;$N(j)$ 表示网页 j 指向的其他网页的数目;$R(i)$ 表示 i 这个网页的权威度;$R(j)$ 表示网页存在指向网页 i 的链接,网页在上一次迭代时的 Page Rank 值;$d(0 < d < 1)$ 是阻尼系数。此算法擅长于计算网页权威度,能够发现权威度比较大的资源,但不能发现主题资源。HITS 算法[8]:如果重要资源在网页子网页内,该算法就很有优势。

1.2 短文本相似度

(1) 基于字符串。设两个文本分别是 $X(x_1,x_2,\cdots,x_n)$ 和 $Y(x_1,x_2,\cdots,x_n)$,则计算文本距离的公式如下:

余弦相似度:

$$\cos\theta = \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}} \tag{2}$$

曼哈顿距离:

$$d_m = \sum_{i=1}^{n} | x_i - y_i | \tag{3}$$

欧氏距离:

$$d_e = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

由于仅是从字面的层次上进行比较,运行速率快,但没有考虑文本中词与词之间的顺序关系、语义关系,所以不能有效地识别近义词。

(2) 基于向量空间。将两个文本编码后看成向量空间[9]中的两个向量,通过余弦距离计算相似度。TF-IDF[10]中的 TF 指的是"词频",IDF 指的是"逆文本频率",公式如下:

$$TF\text{-}IDF(W_i) = tf_j(W_i) \times \log(N/df(W_i)) \tag{5}$$

式中:$tf_j(W_i)$ 表示 W_i 这个词在文档 j 中出现的频率;N 表示的是文档的总数;$df(W_i)$ 表示有多少个文档中出现过 W_i 这个词。通过将所有的文档都进行分析,然后可以得出每一个词的 TF-IDF 值,利用这个值可以对所有文档中每个词进行编码。Word2vec[11]模型分为两种:一种是 CBOW,另一种是 Skip-grim。CBOW 利用词 W_i 前后的各 m 个词预测 W_i,而 Skip-grim 则刚好相反,它利用 W_i 预测前后 m 个词。这两种模型的训练比较相似,这里主要讨论 CBOW。输入层是 $2m$ 个词向量,投影层是 $2m$ 个向量的和。输出层是语料库中所有的词为叶子节点,词的出现次数为权值的哈夫曼树,通过随机梯度下降算法对投影层进行预测,使得条件概率公式 $p(W_i \mid context(W_i))$ 最大化,其中 $context(W_i)$ 指的是 W_i 的上下文 $2m$ 个词。训练结束就得到了每个词的编码。

(3) 基于深度学习。Simase-LSTM[12]模型由 Mueller 等在 2016 年提出。如图 1 所示,模型采用1/ex,因此,输出相似度分数在(0,1]之间,其中,$h_3^{(a)}$ 和 $h_4^{(b)}$ 分别是两个 LSTM 模型的最后隐藏层状态,得到的向量作为特征空间中句子的 embedding,再用距离公式计算相似度。

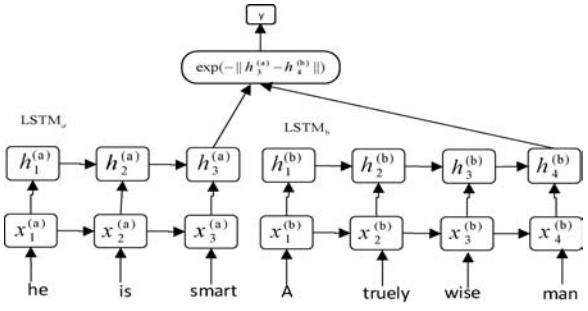

图 1　孪生 LSTM

孪生 CNN[13]模型采用孪生卷积神经网络结构,分别对两个句子进行建模,然后利用一个句子相似度测量层计算文本相似度,最后通过一个全连接层输出相似度得分。其结构如图 2 所示。

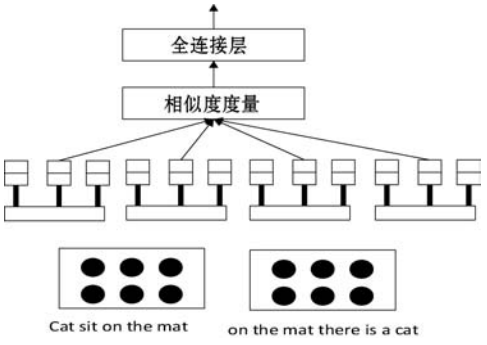

图 2　孪生 CNN

2　设计研究思路

2.1　分布式采集架构

本文系统的采集模块,以一定的种子 URL 开始,通过主题分析以及深度优先搜索、广度优先搜索等策略,确定待采集的 URL。常见的分布式数据采集器框架主要分为控制节点、数据采集节点、存储节点。控制节点包括 URL 管理器、数据存储器、控制调度器。数据采集节点包括 HTML 下载器、HTML 解析器、数据采集调度器。存储节点主要是 MongoDB,负责数据增改删查。数据采集、格式化、存储结束,一定时间后需要对数据更新和判重。系统整体架构如图 3 所示。

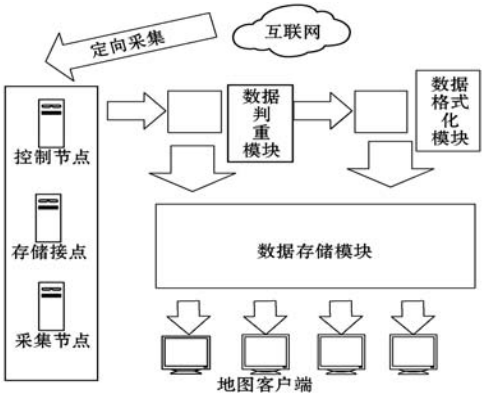

图 3　整体架构设计

网络数据采集功能通过主题网络爬虫实现。与传统网络数据采集不同,采集模块用分布式以及多进程提高采集效率,多台主机可协同采集数据。数据采集节点架构图如图 4 所示。

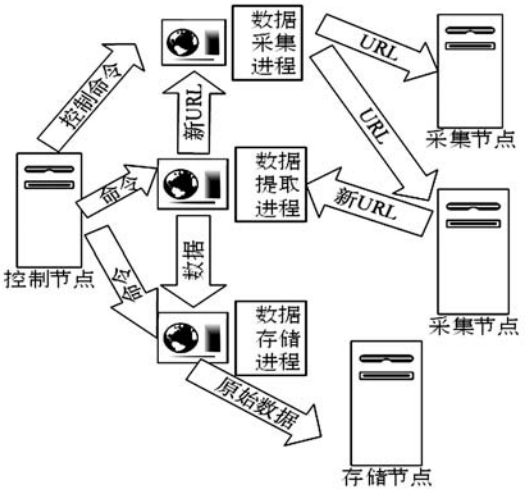

图 4　数据采集节点架构图

1）控制节点:

(1) 控制节点的 URL 管理器。主要包括两个集合,一个集合是已采集 URL 的集合,另一个是未采集的 URL 的集合。如果重复采集 URL,根据 Best first search 算法会出现死循环。

URL 去重复主要有如下解决方案:① 内存去重;② 关系数据库去重、缓存数据库去重。大型成熟的网络数据采集系统采用缓存数据库去重的方式,这样可以尽可能避免内存的限制。URL 管理器是本文系统重要的模块,直接影响着最后数据的召回率。定义一个优先级队列,运用 Page Rank 算法计算优先级,按照优先级大小顺序进行采集。对于上千万条 URL,URL 又非常长,需要做 MD5 处理,以防止内存溢出。MD5 的信息摘要是 128 位,可以减少好几倍的内存消耗。不过 Python 的 MD5 值是 256 位,所以取前 128 位即可。

(2) 控制节点的数据存储器。主要功能是将解析出来的数据,批量放入文本文件或者 HTML 文件。文件名以日期命名,文件格式主要是以 Key-Value 形式保存。

(3) 控制节点的调度控制器。调度控制器主要是产生并启动 URL 管理线程、数据提取进程、数据存储进程,同时维护 4 个队列保持进程间通信。URL_q 队列是 URL 管理进程将 URL 传递给数据采集节点的通道。Result_q 队列是数据采集结点将数据返回给数据提取进程的通道。Conn_q 是数据提取进程将新的 URL 数据提交给 URL 管理进程的通道。Store_q 是数据提取进程将取得的数据交给数据存储进程的通道。

URL 管理进程从 Conn_q 队列获得新的 URL 提交给 URL 管理器,经过去重之后,读取 URL 放入 URL_q 队列中传递给数据采集节点。数据提取进程从 Result_q 队列读取返回数据,并将队列中的 URL 添加到 Conn_q 队列交给 URL 管理进程。数据存储进程从 Store_q 中读取数据并调用数据存储器进行存储。

2) 爬虫数据采集节点:数据采集节点主要包括 HTML 下载器、HTML 解析器、数据采集调度器。执行流程如下:首先数据采集调度器从控制节点中的 URL_q 读取 URL;然后数据采集器先调用 HTML 下载器获得网页的源代码,再从源代码中解析出新的 URL;最后数据采集调度器将新的 URL 和摘要传入 Result_q 队列并交给控制节点。

(1) 数据采集节点的 HTML 下载器。主要利用 HTTP 协议向服务器发送请求,从服务器返回的数据中得到有用的数据。客户端向服务器发出请求,请求建立 TCP 连接。服务器对数据采集器作出响应,并把对应的 HTML 文本发送给数据采集器,然后释放 TCP 连接,数据采集器将 HTML 文本保存。

(2) 数据采集节点的 HTML 解析器。HTML 解析器是通过 Xpath 或者 BeautifulSoup 解析出有用的数据。

(3) 数据采集节点的采集调度器。数据采集节点的调度器需要先连接上控制节点,然后从 URL_q 中获取 URL,下载并解析网页。再把获得的文件交给 Result_q 队列并返回给控制节点。这里需要用到分布式进程——服务进程的创建以及客户进程的创建。

3) 采集模块的存储节点:实现数据持久化存储。

2.2 数据质量管理

断点续传也是网络爬虫设计难点。一个数据采集程序一旦启动就必须等它运行完,如果中途中断就前功尽弃,这样的数据采集程序系统是非常不健壮的。由于不可控因素无法预料,中间会因各种原因中断,如某些数据量大的网站需要采集很长时间,中途可能会遇到断电、网页请求超时、程序崩溃、网页请求频繁被禁止等问题。健壮的网络数据采集系统应该是随时都可以启动,而且每次启动都是从剩下的开始而不是从头开始重复采集。解决该问题的一种策略是记录日志,对于采集失败的 URL 放入日志记录,采集一轮结束后,程序遍历日志记录,再次采集日志记录中采集失败的 URL。如果采用该日志策略,可能会出现某几个 URL 总是采集不成功,程序可能会陷入死循环。另一种策略是 URL 输入两个队列,一个队列存储采集过的 URL,另一个队列存储未采集的 URL,此时就实现了断点续传。断点续传流程设计可使网络爬虫程序可以随时停下,随时启动,并且每次启动都不会做重复工作。

2.3 负载均衡

对于主题爬虫,为了加速主题网站的采集速率,本文系统采集模块改进了常见的网络爬虫请求方法以提高采集效率。不同数据品牌采用多进程,品牌内采用多线程。受 TCP 拥塞控制启发,设计出线程回退算法自动化调整线程数量。该算法描述如下:

(1) 增长阶段:该阶段按照指数规律分配线程的数量。在线程数量指数增长节点,每次分配线程数量结束,在一定的时间窗内检测爬虫是否稳定。剔除检测到的异常的线程,并将下一个时间窗口设置为当前线程数量的一半。

$$N_{t+1} = \begin{cases} 2 \times N_t & state = 1 \\ 1/2 \times N_t & state = 0 \end{cases} \tag{6}$$

式中:N_{t+1} 表示下一个时间窗口的线程数;N_t 表示当前时间窗口的线程数;*state* 表示当前是否异常,值为 0 表示异常,值为 1 表示正常。

(2) 衰减阶段:在该阶段表示以线性规律增加或者减少线程数。经过一定的时间窗口,检测网络爬虫是否异常,正常则线性增加线程数,异常则线性减少线程数。

$$N_{t+1} = \begin{cases} N_t + 1 & state = 1 \\ N_t - 1 & state = 0 \end{cases} \tag{7}$$

2.4 相关算法

(1) RNN。循环神经网络可以将任意长度的序列映射成定长的向量,捕获线性序列的特征非常有效[13],则 RNN 是将 n 个 d_{in} 维向量 $\boldsymbol{X}_{1:n} = \boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n (\boldsymbol{X}_i \in \mathbf{R}^{d_{\text{in}}})$ 序列作为输入返回单个 d_{out} 维向量 $\boldsymbol{y}_n \in \mathbf{R}^{d_{\text{out}}}$ 的函数。

$$\boldsymbol{y}_n = RNN(\boldsymbol{x}_{1:n}) \quad \boldsymbol{x}_i \in \mathbf{R}^{d_{in}} \quad \boldsymbol{y}_i \in \mathbf{R}^{d_{\text{out}}} \tag{8}$$

式(8)隐式定义了:每一个输出向量 $\boldsymbol{y}_i$,可以将返回输出向量序列的函数记为 RNN^*:

$$\begin{aligned} \boldsymbol{y}_{1:n} &= RNN^*(\boldsymbol{x}_{1:n}) \\ \boldsymbol{y}_i &= RNN(\boldsymbol{x}_{1:i}) \\ \boldsymbol{x}_i &\in \mathbf{R}^{d_{\text{in}}} \quad \boldsymbol{y}_i \in \mathbf{R}^{d_{\text{out}}} \end{aligned} \tag{9}$$

构建一个循环神经网络类似于构建一个前馈网络。必须指定输入向量 $\boldsymbol{x}_i$ 和输出向量 $\boldsymbol{y}_i$ 的维度:

$$\begin{aligned} RNN^*(\boldsymbol{x}_{1:n}; s_0) &= \boldsymbol{y}_{1:n} \\ \boldsymbol{y}_i &= o(\boldsymbol{s}_i) \\ \boldsymbol{s}_i &= R(\boldsymbol{s}_{i-1}, \boldsymbol{x}_i) \\ \boldsymbol{x}_i &\in \mathbf{R}^{d_{\text{in}}} \quad \boldsymbol{y}_i \in \mathbf{R}^{d_{\text{out}}} \quad \boldsymbol{s}_i \in \mathbf{R}^{f(d_{\text{out}})} \end{aligned} \tag{10}$$

式中:$o(\cdot)$ 是激活函数;s_i 是隐层单元输出;$R(\cdot)$ 是处理隐层单元的线性函数。加入参数 θ 强调时间步的相同,不同的 R、O 实例会得到不同网络结构,如图 5 所示。

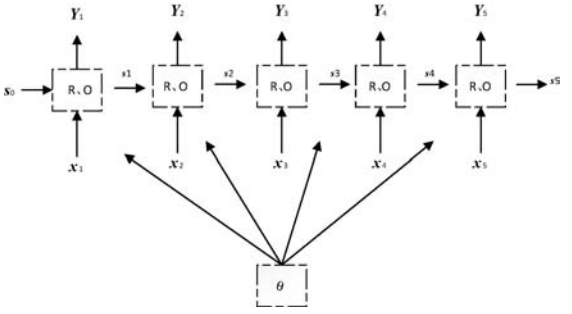

图 5　RNN 图形化表示

(2) 长短记忆网络(LSTM)。长短记忆网络用于解决梯度消失问题[15],并且引入了门结构。将状态向量 $\boldsymbol{S}_i$ 分解成记忆单元和运行单元,记忆单元设计为跨时间记忆并保存相关梯度信息,同时由平滑数学函数控制。形式化定义如下:

$$\begin{cases} \boldsymbol{s}_j = R_{\mathrm{LSTM}}(\boldsymbol{s}_{j-1}, \boldsymbol{x}_j) = [\boldsymbol{c}_j; \boldsymbol{h}_j] \\ \boldsymbol{c}_j = \boldsymbol{f} \odot \boldsymbol{c}_{j-1} + \boldsymbol{i} \odot \boldsymbol{z} \\ \boldsymbol{h}_j = \boldsymbol{o} \odot \tanh(\boldsymbol{c}_j) \\ \boldsymbol{i} = \sigma(\boldsymbol{x}_j \boldsymbol{W}^{xi} + \boldsymbol{h}_{j-1} \boldsymbol{W}^{hi}) \\ \boldsymbol{f} = \sigma(\boldsymbol{x}_j \boldsymbol{W}^{xf} + \boldsymbol{h}_{j-1} \boldsymbol{W}^{hf}) \\ \boldsymbol{o} = \sigma(\boldsymbol{x}_j \boldsymbol{W}^{xo} + \boldsymbol{h}_{j-1} \boldsymbol{W}^{ho}) \\ \boldsymbol{z} = \tanh(\boldsymbol{x}_j \boldsymbol{W}^{xz} + \boldsymbol{h}_{j-1} \boldsymbol{W}^{hz}) \\ \boldsymbol{y}_j = O_{\mathrm{LSTM}}(\boldsymbol{s}_j) = \boldsymbol{h}_j \end{cases} \tag{11}$$

式中:$\boldsymbol{s}_j$ 是时刻 j 的状态;$R_{\mathrm{LSTM}}(\cdot)$ 是 LSTM 算法模型函数;$\odot$ 是 hadamard 乘积操作;$\boldsymbol{W}$ 为 LSTM 模型待更新的权重矩阵。因此,时刻 j 的状态由上一时刻 $\boldsymbol{s}_{j-1}$ 和第 j 时刻的 $\boldsymbol{x}$ 得出。时刻 j 的状态由两个向量组成,分别是 $\boldsymbol{c}_j$ 和 $\boldsymbol{h}_j$。其中,$\boldsymbol{c}_j$ 是记忆组件,$\boldsymbol{h}_j$ 是隐藏状态组件。有三种门结构 $\boldsymbol{i}$、$\boldsymbol{f}$、$\boldsymbol{o}$,分别控制输入、遗忘和输出。门的值由当前输入 $\boldsymbol{x}_j$ 和前一状态 $\boldsymbol{h}_{j-1}$ 的线性组合通过 Sigmoid 激活函数 $\sigma(\cdot)$ 得到。一个更新候选项 $\boldsymbol{z}$ 由 $\boldsymbol{x}_j$ 和 $\boldsymbol{h}_{j-1}$ 的线性组合经过一个非线性激活函数 tanh 得到。

各个值取值范围如下:

$$\begin{aligned} &\boldsymbol{s}_j \in \mathbf{R}^{2 \cdot d_h} \quad \boldsymbol{x}_i \in \mathbf{R}^{d_x} \quad \boldsymbol{c}_j, \boldsymbol{h}_j, \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{o}, \boldsymbol{z} \in \mathbf{R}^{d_h} \\ &\boldsymbol{W}^{xo} \in \mathbf{R}^{d_x \times d_h} \quad \boldsymbol{W}^{ho} \in \mathbf{R}^{d_h \times d_h} \end{aligned} \tag{12}$$

(3) Word2Vec。Word2Vec 有 2 种模型:连续词袋模型(continuous bag-of-words model, CBOW)和 Skip-Gram。本文使用的是 CBOW 模型。

(4) POI 相似度模型。以 Simase-LSTM 设计短文本相似度模型,如图 6 所示。该模型由输入层、词嵌入层、Simase-LSTM 层、全连接层以及输出层组成。将两条 POI 数据的 Name 字段首先进行分词,建立词库字典,以字典中的位置进行初始编号,输入词嵌入层用 Word2Vec 进行计算,得到一组向量。部分词向量效果如表 1 所示。

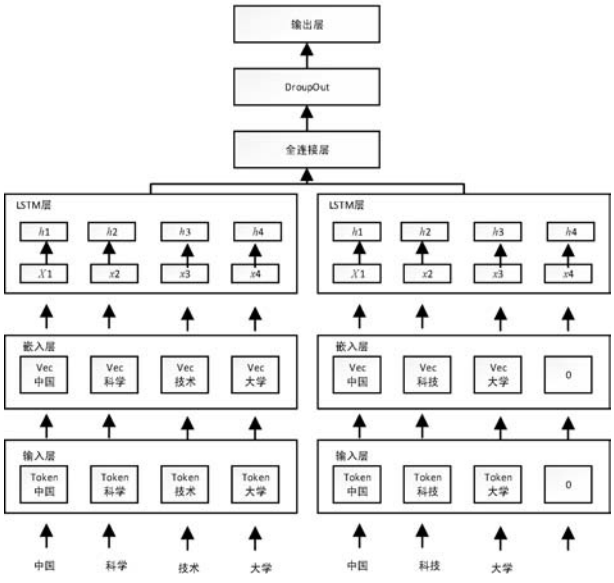

图 6　短文本相似度模型

表 1　部分词向量效果

输入词	近义词	余弦距离
科大	科技大学	0.772
	科技	0.828
	科学技术	0.611
建行	建设	0.788
	建设银行	0.810
	建造	0.601
农行	农业银行	0.872
	农村	0.828
	农忙	0.611
王庄	王庄村	0.976
	王村	0.828
	东王	0.611
安大	安徽	0.656
	安大路	0.989
	安徽大学	0.868

统一长度后输入 Simase-LSTM。学习结果进行拼接。采用简单的单层 LSTM + 全连接层对数据进行训练,两个 LSTM 的输出拼接后作为全连接层的输入,经过 Dropout 和 Batch Normalization 正则化,最终输出结果进行训练。

2.5　POI 数据格式化流程

数据采集之后,需要对数据进行格式化处理,一般 POI 数据的格式是 JSON 形式,这种由多个 Key-Value 形式组成的一条数据便于数据的处理和研究。大多数网页的原始数据都是这种形式。

直接把数据处理程序包含在数据采集模块的实现方式虽然易于实现,但是仔细分析后会发现众多问题。该编程方式增加了程序模块间的耦合度,违反了软件工程的高内聚低耦合的标准,会导致原始数据丢失,比如在数据上线之后数据出现差错,是原始网站的错误还是编程导致的错误无法定位。另外,程序开发完成后,需要对所有数据修改一个属性,如将“经度”与“纬度”合成一个属性“经纬度”,那么就需要一个一个地修改所有的网络爬虫代码,显然这种方式是不明智的。

可以用一种改进的编程方式,即使用设计模式中的工厂模式就可以解决上述所有问题,将“工厂”单独抽象出来一个格式化模块,该格式化模块包含一个数据配置信息映射表和一个映射程序。每一个主题的数据,将是否缺属性、缺哪个属性、属性的映射、原始数据的存储地址都事先写入映射表中,通过该方式将千变万化的数据统一成一种格式,既有利于降低耦合,又有利于批量操作数据,使得系统更加健壮。

图 7 是根据目标分析出的数据格式化模块的程序流程图。

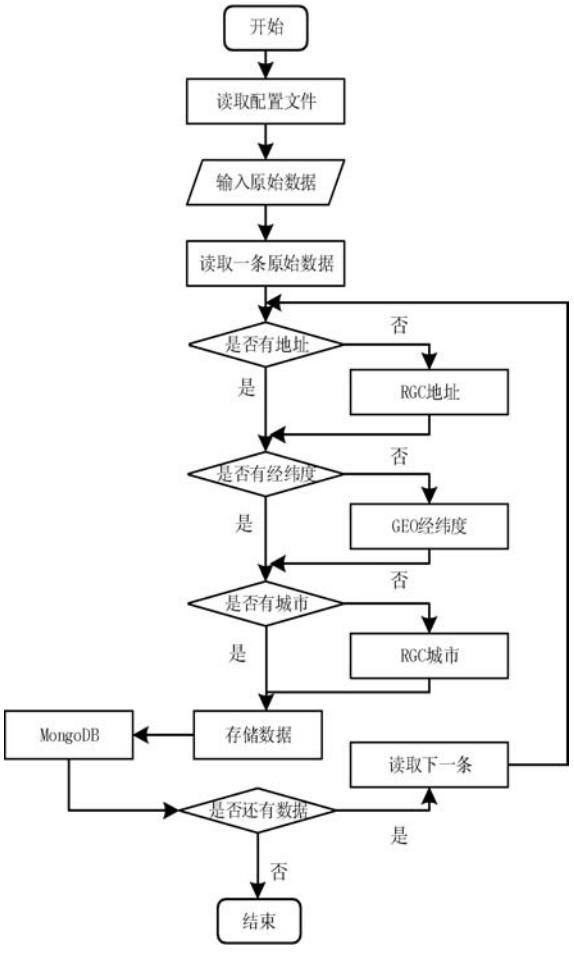

图 7 POI 数据格式化流程图

2.6 POI 数据判重算法

已入库并且已经格式化的数据,经过 3 个月需要进行一次重新的数据采集与判重上线。

得到两批 POI 向量数据后,首先循环遍历第一批数据,对于第一批每个数据,如 OD_i,循环遍历计算第二批数据 ND_j 与第一批数据 OD_i 的距离,找出与 OD_i 距离小于 1 000 米的数据集合 LOD_i,该集合中每条数据的 Shop_name、Address 与 OD_i 的 Shop_name,Address 使用 POI 相似度模型比较并打分,分数最高的一条第二批数据 NDl 即为关联数据。

假设第一批数据数量为 M,第二批数据数量为 N,其中 M 和 N 大小不确定,那么执行判重算法后,关联数据列表 List 的长度一定小于 M 并且小于 N。设关联数据条数为 L,则这 L 条数据既存在于第一批数据,又存在于第二批数据,并且是第一批数据和第二批数据的交集,设为 D。则第一批数据中不在 D 内的数据应下线,第二批不在 D 内的数据应上线。

算法 1 POI 数据判重算法

1. List = [ ];
2. For OD_i←第一批采集数据;
3. For ND_j←第二批采集数据;
4. If Distance(OD_i, ND_j) <1 000:
5. NameScore = Compare(name(OD_i, ND_j));
6. AddressScore = Compare(addre(OD_i, ND_j));
7. TotleScore = 0.8 * NameScore + AddressScore * 0.2
8. 找出与 OD_i 比较后 totleScore 最高的 OD_j。
9. OD_i, OD_j 一起存入 List
10. Return List

3 实验与分析

3.1 分布式系统数据采集性能

为了验证本文方法的可行性,分别对单线程网络爬虫、多线程网络爬虫、多进程网络爬虫,以及本文提出的自适应分布式网络爬虫进行对比实验,表 2 为实验环境的参数。

表 2 实验环境

环境	参数
实验时间	2019 年 7 月 15 日
实验机器	联想 Y470
CPU	I7
网速	500 Mbit/s
内存	16 GB
操作系统	Ubuntu 8

实验将数据品牌“劳力士表”“中国建设银行”“特斯拉汽车”“星巴克咖啡厅”“爱马仕”“小米”“华为”

“学校”等 8 种品牌网站进行编号,如表 3 所示。

表 3　网站品牌

编号	网站
A	劳力士表
B	中国建设银行
C	特斯拉汽车
D	星巴克咖啡厅
E	爱马仕
F	小米专卖店
G	华为专卖店
H	北京便民政务网站小学模块

数据采集实验:将本文提出的负载均衡自适应算法应用到多线程以及本文设计的分布式爬虫系统上,得到如图 8 所示的实验结果。

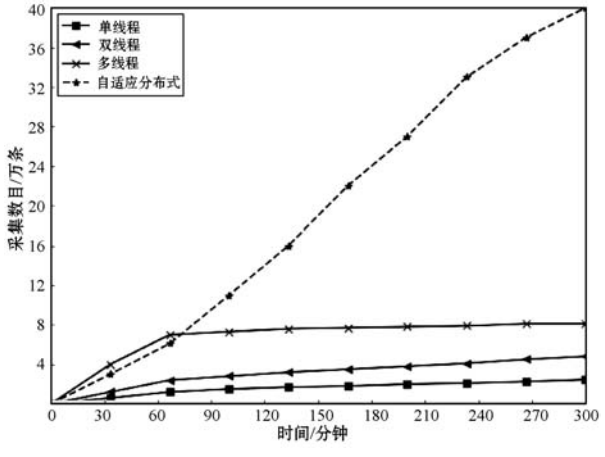

图 8　POI 数据采集性能对比图

在刚开始采集阶段,多线程的采绩效率明显高于其他方案,在采集 60 分钟之后,由于源网站反趴策略的限制,效率开始下降。自适应分布式采集策略趋于稳定。

3.2　POI 数据的短文本相似度算法

(1) 数据集的构造。将从表 3 中采集到的 8 类品牌数据随机取出 3 000 条,标注是否相似,1 代表相似,0 代表不相似。样本示例如表 4 所示。

表 4　实验环境

名称 1	名称 2	相似性
北京大学	北京科技大学	0
中科大	中国科学技术大学	1
招商银行上地	海淀区上地路招商银行	1
小米专卖店蜀山	蜀山区小米专卖店	0
星巴克合肥店	合肥市星巴克咖啡厅	1

(2) 模型训练。把 2 批 6 000 条数据按照 7∶2∶1 分为训练集、验证集、测试集。使用梯度下降法更新神经网络权值。batch_size 是每块训练样本数,本文系统中为 15,每次对 15 个参数进行训练。Epochs 参数是迭代次数,本文系统中为 100。

如图 9 所示,模型的训练损失每轮都在下降,准确率每轮都在上升,达到 80 轮之后,呈现稳定状态。准确率达到 98%,损失函数降至 1.3%。

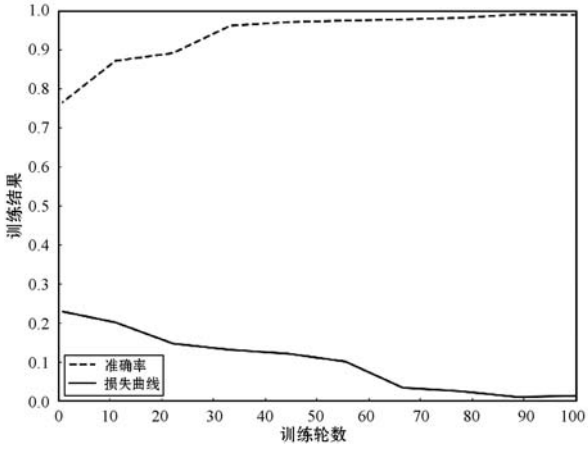

图 9　模型训练结果

(3) 模型验证。如图 10 所示,达到 80 轮之后,模型在验证集上准确率达到 95.2%,损失函数值降至 17%。

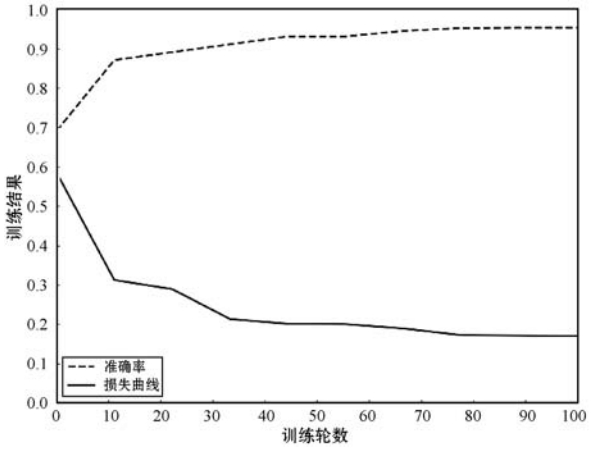

图 10　模型验证结果

(4) 模型测试。除了使用本文介绍的 Word2Vec 结合 LSTM 方法,实验还对比了 Word2Vec 结合 CNN 方法、仅用 Word2Vec 方法。以下简称本文算法、WV-CNN、WV-Cosine。其中 WV-Cosine 指的是将短文本用 Word2Vec 编码后直接用 COS 函数计算短文本相似度。各方法在测试集上的性能如表 5 和图 11 所示。

表 5　多种算法在测试集上的准确率比较

方法	测试组数	正确组数	准确率
WV-Cosine	600	414	0.685
WV-CNN	600	513	0.885
本文算法	600	561	0.935

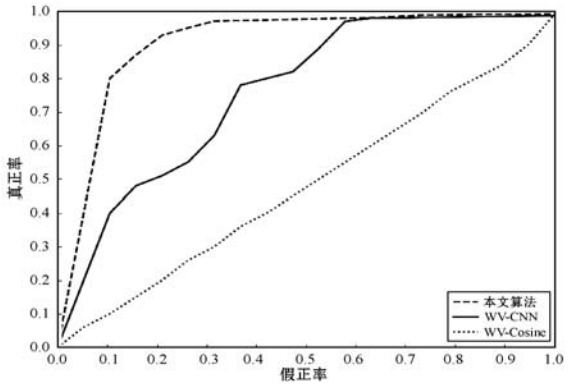

图 11 模型 ROC 曲线

本文模型在准确率和 ROC 曲线[16]上的表现优于其他算法。WV-Cosine 算法效果最差,因为 Word2Vec 虽然解决了词语间语义关系的问题,但是其类似于词袋模型,简单地把词语向量堆积组成文本向量,往往会造成误判。虽然 WV-CNN 有较高准确率,但不适用于序列化文本的处理。对于文本处理 LSTM 更有优势。由于计算的相似度是介于[0,1]之间的浮点数,因此需要规定阈值,这里定义相似度小于 0.8,算法判断为不相似,相似度大于 0.8,算法判断为相似。这样就可以计算真正率与假正率。

从图 11 中 ROC 曲线看出,本文提出的短文本相似模型具有最佳性能。

3.3 基于短文本相似度的 POI 判重算法

(1) 判重算法的基本流程。三个月前采集一批数据,现在采集一批数据,为了保证电子地图实时更新,需要将第一批没有的数据识别出之后上线。两批数据的交集不用上线。基于短文本相似度的 POI 判重算法的目的就是找到两批数据的交集。流程如图 12 所示。

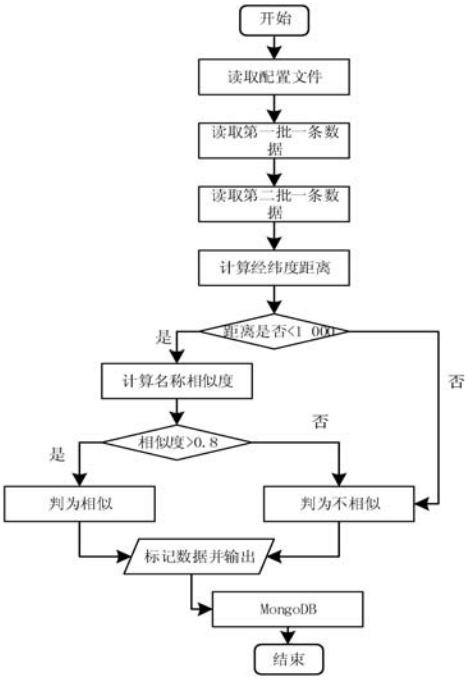

图 12 POI 数据判重算法

(2) 判重算法性能比较。将短文本相似度算法分别换成 WV-CNN、WV-Cosine,与本文的算法性能对比如图 13 所示。

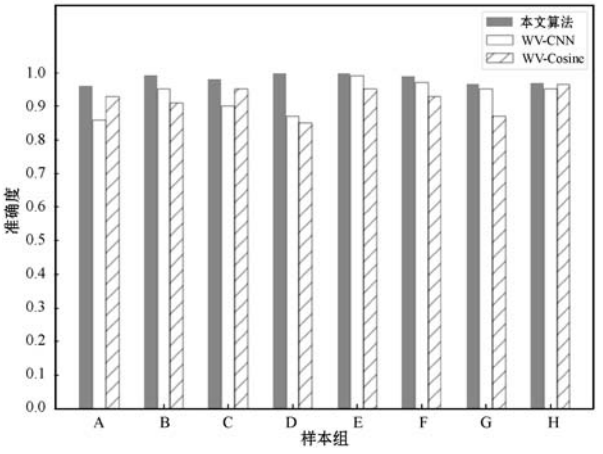

图 13 POI 数据判重算法性能对比

由图 13 可得,本文提出的算法更具有稳定性,在各类品牌上的准确率均保持在 95% 以上。

3.4 系统实验实例

系统采用自适应分布式爬虫技术从银行、小米手机店、汽车 4S 店等官方网站采集电子地图的 POI 数据,经格式化模块补全属性后进行上线。经过一定时间重新采集数据、判重上线。后端使用 MongoDB 存储数据。主要功能包括数据抓取、数据格式化、数据判重、数据入库。腾讯地图、高德地图等地图厂商可使用本文思想自动化更新数据。以下是系统实现后得到的数据例子:

(1) 原始数据例子。从魅族手机官方网站抓取的一条营业厅的数据。

{"province": "四川省", "name": "四川宜宾授权服务体验中心", "city": "宜宾市", "address": "四川省宜宾市翠屏区东街经纬商场 9 楼 8 号"}

(2) 格式化数据的例子。由于原始数据可能会缺少字段,如原始数据例子中没有经纬度信息,在格式化模块需要用百度地图工具 API 将"四川省宜宾市翠屏区东街经纬商场 9 楼 8 号"这条地址信息转换成经纬度信息,并加上唯一 ID 号。

{"province": "四川省",
"thirdId": "3 326 575.453 603 036",
"name": "四川宜宾授权服务体验中心",
"city": "宜宾市",
"longitude": "11 648 275.596 548 6",
"address": "四川省宜宾市翠屏区东街经纬商场 9 楼 8 号",
"latitude": "3 326 575.453 603 036"}

这样就可以将这条数据上线到腾讯地图 App 中。

(3) 数据的判重例子。数据上线到腾讯地图 App 中,3 个月以后,腾讯地图 App 的数据上线人员再次运行爬虫程序,抓取同样的数据,发现原始数据“name”字段有变化,如:

三月前:"name": "四川宜宾授权服务体验中心"。

现在:"name": "魅族官方授权服务体验中心(四川宜宾店)"。

可能经纬度也有变化,上线人员可以认为这个营业厅可能搬至别处。因此,可以运行本系统的判重算法,通过经纬度计算,得到此时 POI 数据与三月前同一 POI 数据距离"distance": 65.939 936 304 486 3,即 65 米。旧 name(即"oldname": "四川宜宾授权服务体验中心")与新 name(即"name": "魅族官方授权服务体验中心(四川宜宾店)")相似度分数为"score": "0.731 213"。因此可以判断该条数据与三个月前那条数据是同一条数据,仅让新采集的未找到关联的数据上线即可。

{"province":四川省",
"distance": 65.939 936 304 486 3,
"sametype": "diff",
"thirdId": "3 326 575.453 603 036",
"oldname": "四川宜宾授权服务体验中心",
"city": "宜宾市",
"longitude": "11 648 275.596 548 6",
"score": "0.731 213",
"name": "魅族官方授权服务体验中心(四川宜宾店)",
"address": "四川省宜宾市翠屏区东街经纬商场 9 楼 8 号",
"latitude": "3 326 575.453 603 036",
"guid": "fd3fcb4e-a2a4-11e7-a06b-6c92bf270e68"}

4 结　语

本文提出并设计的系统很好地解决了地图 App 的数据来源问题,全程自动化处理,具有很好的用户体验。但在以下方面也有值得改进的地方:

(1) 运行效率。虽然本文系统在目前来看,运行速度还算可以,但是数据量如果加大,效率还有待提高。由于本文系统是面向过程设计的整体架构,因此,一个功能模块不稳定,就会让后期数据处理无法执行。

(2) 算法的准确率。NLP 的文本相似度算法的准确率一直在 0.93 左右,所以算法准确率的提高有待进一步研究。

参 考 文 献

[1] 刘景发,李新,蒋盛益. 基于网页空间进化算法的暴雨灾害主题爬虫策略[J]. 计算机工程,2019,45(2):184 - 190.

[2] 萧婧婕,陈志云. 基于灰狼算法的主题爬虫[J]. 计算机科学,2018,45(S2):146 - 148,166.

[3] Morrison D R,Sauppe J J,Zhang W,et al. Cyclic best first search:using contours to guide branch-and-bound algorithms [J]. Naval Research Logistics,2017,64(1):64 - 82.

[4] Cho J,Garcia-Molina H,Page L. Efficient crawling through URL ordering[J]. Computer Networks and ISDN Systems,1998,30(1 - 7):161 - 172.

[5] Filho C J A B,Neto F B D L,Lins A J C C,et al. A novel search algorithm based on fish school behavior[C]//2008 IEEE International Conference on Systems,Man and Cybernetics,2008.

[6] 杨仁广,宋宇,孟祥增. 一种改进 Shark-Search 的多媒体主题搜索算法[J]. 计算机工程与应用,2010,46(14):152 - 154.

[7] 库珊,刘钊. 基于 PageRank 与 HITS 的改进算法的网页排名优化[J]. 武汉科技大学学报,2019,42(2):155 - 160.

[8] 刘昊,洪宇,姚亮,等. 基于 HITS 算法的双语句对挖掘优化方法[J]. 中文信息学报,2017,31(2):25 - 35.

[9] 申强强,熊泽宇,熊岳山. 一种新的基于段向量的文本自动摘要方法[J]. 计算机工程与科学,2019,41(6):1064 - 1070.

[10] Nurdiansyah Y,Andrianto A,Kamshal L. New book classification based on Dewey Decimal Classification (DDC) law using tf-idf and cosine similarity method[J]. Journal of Physics:Conference Series,2019,1211:012044.

[11] Vargas-Calderón V,Camargo J E. Characterization of citizens using word2vec and latent topic analysis in a large set of tweets[J]. Cities,2019,92:187 - 196.

[12] Mueller J,Thyagarajan A. Siamese recurrent architectures for learning sentence similarity[C]//Thirtieth AAAI Conference on Artificial Intelligence,2016.

[13] Hughes L H,Schmitt M,Mou L,et al. Identifying corresponding patches in SAR and optical images with a pseudo-siamese CNN[J]. IEEE Geoscience and Remote Sensing Letters,2018,15(5):784 - 788.

[14] 鄂海红,张文静,肖思琪,等. 深度学习实体关系抽取研究综述[J]. 软件学报,2019,30(6):1793 - 1818.

[15] 王竣平,白宇,蔡东风. 采用 BI-LSTM-CRF 模型的数值信息抽取[J]. 计算机应用与软件,2019,36(5):138 - 144.

[16] Cao R,López-de-Ullibarri I. ROC curves for the statistical analysis of microarray data[M]//Microarray Bioinformatics,2019:245 - 253.