计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(6):48−55 [doi: 10.15888/j.cnki.csa.008584]

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

基于关键词生成的网格事件相似度并行计算①

陈　钢, 陈健鹏, 佘祥荣, 秦加奇, 陈　剑

(长三角信息智能创新研究院, 芜湖 241060)
通信作者: 陈　钢, E-mail: cheng@ustc.win

摘　要: 为实现在海量网格事件库中快速、准确地检索事件, 本文提出一种基于关键词生成的网格事件相似度并行计算方法. 该方法通过双向 LSTM 网络的编码器和单向 LSTM 网络的解码器构建指针生成网络生成事件关键词, 使用记忆网络作为指针生成网络的序列信息存储单元, 并将注意力机制用在输入序列上以将更重要的信息输入至解码器, 同时引入覆盖机制来解决生成重复文本问题. 在生成事件关键词后, 基于结构相似度和情境相似度计算事件总体相似度, 并利用 GPU 对 LSTM 网络和相似度计算进行加速. 实验结果表明: 相比基于机器学习的计算方法, 该方法在事件相似度计算性能上更好, 最高获得了 4.04 倍的加速比.

关键词: 关键词生成; 网格事件; 相似度计算; 指针生成网络; 图形处理器; 并行计算

引用格式: 陈钢,陈健鹏,佘祥荣,秦加奇,陈剑.基于关键词生成的网格事件相似度并行计算.计算机系统应用,2022,31(6):48–55. http://www.c-s-a.org.cn/1003-3254/8584.html

Parallel Calculation of Grid Event Similarity Based on Keyword Generation

CHEN Gang, CHEN Jian-Peng, SHE Xiang-Rong, QIN Jia-Qi, CHEN Jian

(Yangtze River Delta Information Intelligence Innovation Research Institute, Wuhu 241060, China)

Abstract: For quick and accurate retrieval in massive grid events, this study proposes the parallel similarity calculation of grid events based on keyword generation. The method generates grid event keywords through the pointer-generator network based on the encoder of a bidirectional LSTM network and the decoder of a unidirectional LSTM network. It uses a memory network to store sequence information and applies the attention mechanism to the input sequence, enabling more important information into the decoder. It also introduces the overwriting mechanism to avoid duplicate texts. After the keywords are generated, the overall similarity is calculated based on the structural similarity and situational similarity of events. GPU is utilized to accelerate the LSTM network and similarity calculation. Experimental results show that the method has better performance in similarity calculation than methods based on machine learning, with a speedup ratio reaching 4.04 times.

Key words: keyword generation; grid event; similarity calculation; pointer-generator network (PGN); graphics processing unit (GPU); parallel computing

社会治理网格化将城市按照一定标准划分成一系列网格单元, 网格化系统所沉淀的海量事件数据能够全面、及时地反映城市问题. 在城市应急管理中, 基于历史网格事件库的案例推理[1] 对于科学救援、精准决策具有重要意义. 事件检索是案例推理的关键步骤, 通过在网格事件库中检索出与目标事件相似的历史事件(集), 进而辅助目标事件的处置决策. 网格事件库中的数据量越庞大, 经验积累越充分, 对决策支持的力度也

① 基金项目: 2021 年安徽省重点研究与开发计划 (202104a05020071); 2021 年安徽省科技创新战略与软科学研究 (202106f01050056)
收稿时间: 2021-09-07; 修改时间: 2021-10-11, 2021-10-19; 采用时间: 2021-11-26; csa 在线出版时间: 2022-05-26

越大. 从相似的事件处置中得到经验性的知识, 必须在合理时间内在网格事件库中找到相似事件[2]. 因此, 高性能网格事件相似度计算决定了案例推理的效用. 近年来, 图形处理单元 (graphics processing unit, GPU) 在硬件架构上取得了长足的进步. 片上运算单元密集、存储带宽高效等特点使得 GPU 非常适用于数据相关性较低的大规模并行计算. 事件相似度计算对大量事件数据进行相同处理, 其中蕴含丰富的数据并行性, 适合在 GPU 上加速执行[3].

事件关键词提取大多数通过 TF-IDF 提取候选关键词, 利用 Word2Vec 计算词向量, 并采用特征工程对候选关键词进行特征提取, 再经由支持向量机、决策树等算法将关键词提取转换为二分类问题[4]. 这种方法需要做大量的特征工程, 特征的选取和分析方式复杂, 还可能会造成前端特征与后端任务的脱节. 此外, 词向量加机器学习的方法仅能够基于给定的事件文本提取关键词, 无法挖掘网格事件中蕴含的重要特征. 指针生成网络 (pointer-generator network, PGN) 结合了传统 Seq2Seq 模型和指针网络的优势, 在生成新词的同时也具备了从原文复制单词的能力, 允许模型从源文本中复制词用作生成词来解决词表无法覆盖 (out of vocabulary, OOV) 的问题, 并引入了覆盖机制以改善生成新词时的重复问题, 提高模型的表达能力[5]. 为了在海量网格事件库中快速、准确地检索事件, 本文提出一种基于关键词生成的网格事件相似度并行计算方法, 具体创新点如下:

(1) 引入记忆网络 (memory network) 对基于 LSTM 网络的 PGN 进行改进, 用以增强其记忆能力;

(2) 针对 GPU 体系结构特点, 对 LSTM 网络的计算过程进行优化;

(3) 提出基于历史相似事件的先验知识来计算事件相似度阈值的方法;

(4) 通过实验证明本文网格事件相似度技术方法的有效性.

1 相关研究

文献 [6] 设计了一种基于网格事件大数据汇总共享、通过数据梳理及案例推理技术实现科学决策的方法, 并基于这一方法设计了网格化管理辅助决策支持系统. 文献 [7] 通过文本分词、特征词提取、基于情景相似度的突发事件情报感知方法, 结合当前事件演化态势和以往的经验性知识实现对突发事件的识别和预判. 文献 [8] 在传统词嵌入模型中增加了 Ngram 和汉字语义信息并与知网融合, 在此基础上将 WNCH 方法应用到文本属性相似度的计算. 文献 [9] 通过扩展 Needleman-Wunsch 算法的得分函数以结合时间、空间信息, 通过粒度调控实现了从不同的粒度来计算时空事件序列的相似度. 文献 [10] 对突发事件进行情景要素分解, 引入支持向量机和相似度算法, 基于粒度原理构建了一种融合情景的动态响应模型. 文献 [11] 提出一种融合句法特征和句法相似度的网络舆情突发事件识别方法. 文献 [12] 提出了食品安全事件的多层多级语义结构排序策略算法, 计算食品安全数据与语义结构模板的相似度, 确定其综合得分, 选择适当的阈值确定食品安全事件精度.

尽管上述方法能够较好地完成案例检索, 但案例检索的事件相似度计算方面存在局限性, 大多数研究基于案例的数值属性、模糊属性、符号属性进行事件之间的相似度计算, 忽略了事件的文本属性, 如事件原因、事件摘要等, 这些属性对于案例检索都是不可或缺的[13]. PGN 主要解决传统 Seq2Seq 模型中输出严重依赖输入的问题, 突破了模型输出端对词汇表长度的限制, 在摘要生成任务中具有良好的表现. 文献 [14] 针对生成式文本摘要应用场景, 提出了以 Transformer 为基础的摘要模型, 并在 Transformer 模型中加入了指针生成网络和覆盖损失进行优化. 文献 [15] 针对代码注释自动生成引入指针生成网络模块, 在解码的每一步实现生成词和复制词两种模式的自动切换, 以此解决无法生成 OOV 词的问题.

2 网格事件相似度计算

网格事件可以分为城市管理、环境保护、物业管理、平安建设、应急突发事件等类型, 包括基本属性 (事件标题、等级、类别, 发生地点、发送时间等信息) 和情景属性 (事件原因、对象、发生环境、应对任务等信息). 在计算事件相似度之前, 需要生成事件关键词以确定该事件的基本属性和情景属性.

2.1 关键词生成

长短期记忆网络 (long short-term memory, LSTM) 是在循环神经网络 (recurrent neural network, RNN) 基础上改进而来的一种神经网络模型, 可以解决长期依赖问题. LSTM 神经网络使用输入门、忘记门和输出

门 3 种门结构以保持和更新细胞中信息的增减. 然而, 单向 LSTM 只能处理一个方向上的信息, 无法处理另外一个方向上的信息, 双向 LSTM (bidirectional LSTM, BiLSMT) 通过双向语义编码结构获取上下文信息, 能够更好地对网格事件信息进行提取[16], 而注意力机制 (attention) 可以更深层次地挖掘网格事件不同类型的关键词. 本文提出的基于 PGN 的网格事件关键词生成模型如图 1 所示.

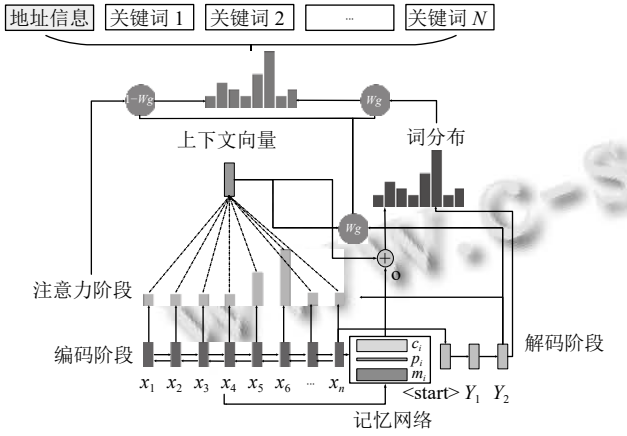

图 1 基于指针生成网络的关键词生成

2.1.1 编码器

在网格事件特征提取中, 当前时间步长的隐藏状态与前一时刻和下一时刻相关联. BiLSTM 使用两个不同方向的 LSTM 分别从网格事件文本的前端和后端进行遍历, 利用两个并行通道, 既能获得正向的累积依赖信息, 又能获得反向的未来的累积依赖信息, 提取的特征信息更加丰富. 因此, 编码器部分采用 BiLSTM 网络, 如图 2 所示.

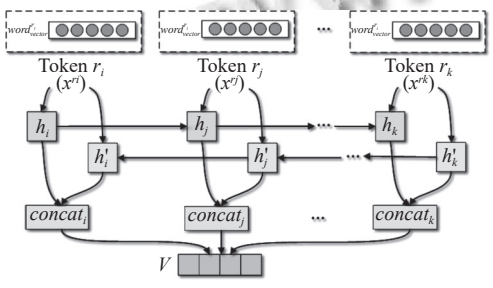

图 2 BiLSTM 结构

对于网格事件候选词序列$X=\{x_1,x_2,\cdots,x_n\}$ (n 为输入序列的长度), 按照顺序先输入到 Embedding 层, 将候选词映射到高维向量上, 然后再将处理好的序列输入到编码器中, 得到每个序列的隐藏状态集合$E=\{e_1,e_2,\cdots,e_n\}$. 对第 i 个隐藏状态 e_i 来说, 由于采用的是 Bi-LSTM 网络, 算法会从前往后和从后往前两个方向计算, 得到的隐藏状态 e_i 会充分关联上下文信息. BiLSTM 计算过程如式 (1) 到式 (6) 所示. 首先, 遗忘门根据上一个记忆单元的输出 h_{t-1} 和输入数据 x_t 产生一个 0–1 之间的数值 f_t 来决定上一个长期状态 C_{t-1} 中信息丢失多少. h_{t-1} 和 x_t 通过输入门确定更新信息得到 i_t, 同时通过一个 tanh 层得到新的候选记忆单元信息 C'_t. 通过遗忘门和输入门的操作, 将上一个长期状态 C_{t-1} 更新为 C_t. 最后, 由输出门得到判断条件, 然后通过一个 tanh 层得到一个 (–1, 1) 之间的值 o_t, 该值与判断条件相乘来决定输出当前记忆单元的哪些状态特征.

$$f_t=\sigma(W_f\cdot[h_{t-1},x_t]+b_f) \tag{1}$$

$$i_t=\sigma(W_i\cdot[h_{t-1},x_t]+b_i) \tag{2}$$

$$C'_t=\tanh(W_C\cdot[h_{t-1},x_t]+b_C) \tag{3}$$

$$C_t=f_t*C_{t-1}+i_t*C'_t \tag{4}$$

$$o_t=\sigma(W_o\cdot[h_{t-1},x_t]+b_o) \tag{5}$$

$$h_t=o_t*\tanh(C_t) \tag{6}$$

2.1.2 解码器

解码器部分采用单向 LSTM 结构, 假设解码器的输入为$Y=\{y_1,y_2,\cdots,y_m\}$ (m 为解码器输入序列个数), 在解码过程中输入序列按照顺序先输入到 Embedding 层, 将输入序列映射到高维向量上, 得到每个输入序列的词向量表示, 然后再将处理好的序列输入到解码器中, 得到每个序列的隐藏状态集合$D=\{d_1,d_2,\cdots,d_m\}$. 为了使解码器产生更合适的输出, 本文将注意力机制用在输入序列上以将更重要的信息输入至解码器. 同时, 引入覆盖机制来解决 Seq2Seq 模型生成重复文本的问题[17]:

$$c^t=\sum_{t'=0}^{t-1}\alpha^{t'} \tag{7}$$

$$u_i^t=v^{\mathrm{T}}\tanh(W_1e_i+W_2d_t+W_cc_i^t+b) \tag{8}$$

$$\alpha^t=Softmax(u^t) \tag{9}$$

其中, c_t 是先前时间步的注意力权重叠加后得到的覆盖向量. 对上下文向量 u_t 进行 $Softmax$ 激活函数操作后, 得到的结果视为在输入序列元素上的概率分布, 其中 v^{T}、W_1、W_2、W_c 和 b 是模型学习参数.

PGN 使用 LSTM 作为编码器和解码器, 但 LSTM 的隐藏状态记忆存储能力有限, 无法存储太多的信息, 且容易丢失一部分语义信息. 为此, 本文通过引入记忆网络存储序列信息对其进行改进以增强其记忆能力. 在图 1 中, m_i 是根据单词的词向量和位置形成的句子向量, 其公式为:

$$m_i = \sum_j l_j \cdot X_{ij} \tag{10}$$

其中, X_{ij} 是句子中每个单词的词向量矩阵, l 记录每个单词的位置信息, l 计算公式为:

$$l_{kj} = (1 - j/J) - (k/d)(1 - 2j/J) \tag{11}$$

其中, J 是句子长度, d 是编码维度, 经过 *Softmax* 之后形成一个权重向量 P:

$$p_i = Softmax\left(h_t^{\mathrm{T}} m_i\right) \tag{12}$$

文本向量 c_i 由权重向量 P 和 m_i 构成:

$$c_i = p_i m_i \tag{13}$$

记忆网络的最终输出计算公式为:

$$o = \sum_i p_i c_i \tag{14}$$

在时刻 t 的输入序列表征可以为$e_t = \sum \alpha_i^t e_i$, 将 e_{t} 和 d_{t} 经过两个线性层得到单词表分布:

$$P_v(W) = Softmax(V'(V[(e_t + o), d_t] + b) + b') \tag{15}$$

权重 w_{g} 的计算方式为:

$$w_g = \sigma(W_1' e_t + W_2' d_t + W_3' x_t + b) \tag{16}$$

其中, W'_1、W'_2、W'_{c} 和 b 是模型学习参数. 经过扩充之后的单词表, 在时间步 t 的词概率分布为:

$$P(W) = w_g P_v(W) + (1 - w_g) \sum \alpha_i^t \tag{17}$$

2.1.3 模型训练

在 PGN 中 t 时刻模型损失值可通过对目标输出单词 y_{t} 和覆盖向量计算覆盖损失求得:

$$loss_t = -\log P(y_t) + \lambda \sum \min(\alpha_i^t, c_i^t) \tag{18}$$

$$L_{\mathrm{event}} = \frac{1}{T} \sum_{t=0}^{T} loss_t \tag{19}$$

根据输出端的概率分布, 得到输入序列概率最高的 L 个候选词作为网格事件的关键词.

2.2 相似度计算

计算网格事件相似度需要从结构相似度 (属性相似程度) 和情景相似度 (关键词相似程度) 两方面来综合衡量.

2.2.1 结构相似度

结构相似度比较两个网格事件具有的共同属性, 共同属性越多, 事件相似程度也越高.

$$SIMS(A, B) = \frac{2C}{N_A + N_B} \tag{20}$$

其中, A 为新发送事件, B 为网格事件数据库中历史事件, C 为两个事件的共同属性数, N_A 和 N_B 分别为事件 A 和事件 B 的属性数.

2.2.2 情景相似度

在生成网格事件关键词后, 利用这些关键词与网格事件库中已构建的事件情景 (即一系列事件关键词所组成的集合) 进行相似度计算. 假设 $SIMC(A, S)$ ($0 \leqslant SIMC(A, S) \leqslant 1$) 表示包含 m 个关键词的事件$A = \{v_1, v_2, \cdots, v_m\}$与包含 n 个关键词的事件情景$S = \{w_1, w_2, \cdots, w_n\}$之间的相似度: 若 $SIMC(A, S)$ 低于某一阈值, 则表明事件 A 和事件情景 S 相似度较低 (即不具有相似度); 若 $SIM(A, S)$ 高于某一阈值, 则表明事件 A 和事件情景 S 相似度较高 (即具有相似性). 本文根据网格事件库中历史相似事件的先验知识来设定事件相似度阈值. 假设网格事件库中有 N 个事件, 划分为 M 种类型, 每个类型下相似事件的数量为 S_i (i=1,⋯, M), 则采用如下步骤来确定相似度阈值:

步骤 1. 对于事件类型 k 中标记为相似的网格事件两两计算其相似度 $SIMC(x, y)$, 取其平均值得到 ave_k;

步骤 2. 对于所有 M 种类型的网格事件重复步骤 1;

步骤 3. 根据每种类型事件的占比权重由$\{ave_1, ave_2, \cdots, ave_M\}$确定最终的阈值.

用于揭示概念与概念之间以及概念所具有的属性之间的关系的知网 (HowNet) 提供了汉语词语相似度的计算方法[7], 本文借鉴 HowNet 方法来计算事件关键词与事件情景之间的相似度. 假设关键词 W_1 可能包含 P 个概念$\{c_1, c_2, \cdots, c_P\}$, 关键词 W_2 可能包含 Q 个概念$\{d_1, d_2, \cdots, d_Q\}$, 这些概念之间的最大相似值就是 W_1 和 W_2 的相似度. 因此, 事件 A 和事件情景 S 的相似度计算公式为:

$$SIMC(A, S) = \frac{1}{m \times n} \sum_i \sum_i SIM(c_i, d_j) \tag{21}$$

2.2.3 总体相似度

两个网格事件的总体相似度由结构相似度和情景

相似度构成:

$$SIM(A,B)=\alpha SIMS(A,B)+\beta SIMC(A,B) \quad (22)$$

其中, α 为结构相似度的权重, β 为情景相似度的权重, 且 $\alpha+\beta=1$. 根据网格事件数据库中历史事件相似度情况, 利用 Viterbi 算法求解出 α 为 0.19、β 为 0.81. 可见, 情景相似度总体上决定了事件相似度.

2.3 GPU 并行计算

在基于 PGN 生成事件关键词时, LSTM 网络涉及大量的矩阵和向量运算. 此外, 计算网格事件库中两两事件的相似性的计算复杂度非常高, 尤其是存在情景相似度的计算时. 因此, 本文利用 GPU 对这两个过程进行加速.

2.3.1 LSTM 网络加速

由于矩阵乘操作占了 LSTM 网络计算过程的 95% 时间, 可以从权重矩阵列内 (输入向量按列与权重矩阵相乘)、列间并行 (每个输入值与权重矩阵内的每一列元素进行乘法计算) 以及权重矩阵之间并行 (每个输入值与门单元权重矩阵进行向量矩阵乘法运算) 对 LSTM 网络进行加速计算. 由于 LSTM 网络不同层间具有强依赖关系, 需要先完成底层计算, 才能对上层计算. 因此层与层之间无法并行处理, 只能在层内并行计算. 此外, 不同时间步之间也具有强依赖关系, 所以只能在每个时间步内的每层做并行计算, 图 3 展示了 LSTM 网络的可并行性.

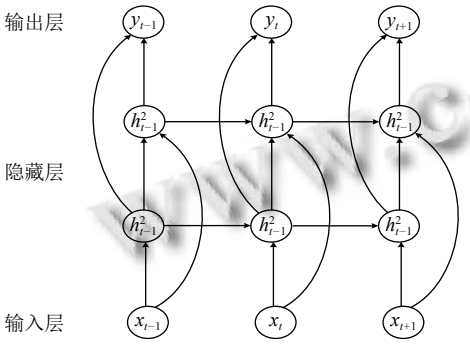

图 3 LSTM 网络可并行性

在 LSTM 网络中, 存在 3 种不同形式的矩阵乘运算: 矩阵向量乘 (MMV)、向量矩阵乘 (VMM) 和向量外积 (VEP). MMV 出现在前向计算中, VMM 和 VEP 出现在反向计算中. GPU 核心 (kernel) 程序启动会导致开销, LSTM 网络每个单元有 3 个门, 每个门具有两个不同的权值矩阵, 若不加以合并, 完成一次前向和一次后向计算分别需要启动 12 次 kernel, 导致开销较大. 权值矩阵过小无法充分利用 GPU 并行计算优势, 为此本文实施了两个层次的权值合并.

(1) 列向合并

列向合并针对同一层相同门的不同权值进行合并, 因此输入数据根据扩充顺序同步进行合并. 以输入门为例, 输入门接收的输入包括从当前时间步的输入 x_t 与上一个时间步的输出 h_{t-1}, 其对应的权值分别为 W_t 和 W_{t-1}, 合并后的权值 $W_{\text{com}}=[W_t, W_{t-1}]$, 在行数不变的基础上对列做了加和. 对于输入合并, 合并后的输入 $\boldsymbol{I}_{\text{com}}=[x_t, h_{t-1}]$, 对应列也做了加和:

$$W_{\text{com}} * I_{\text{com}} = [W_t * x_t, W_{t-1} * h_{t-1}] \quad (23)$$

相同门的前向计算和后向计算过程是相互对应的, 因此前向和后向的计算公式完全一样. 值得注意的是, 后向过程原本对 x 和 h 分别进行更新, 现在对合并后的向量进行更新.

(2) 行向合并

在对行向不同门的权值进行合并的过程中, 输入数据不需要进行变换. 将 f、i、o 三个门合并后得到新权值矩阵 W_{join}, 其行数变为原来的 3 倍, 列数不变. 行向合并计算过程如下:

$$W_{\text{join}} * I_{\text{com}} = (W_f * I_{\text{com}}, W_i * I_{\text{com}}, W_o * I_{\text{com}})^{\text{T}} \quad (24)$$

值得注意的是, 由于后向计算过程中不同门的激活函数不同, 导致各个门的梯度计算不完全一致, 因此, 需要分别计算各个门的梯度系数.

通过对权值进行列向和行向合并, LSTM 网络进行一次前向计算仅需要启动 2 次 kernel, 进行一次后向计算仅需要启动 3 次 kernel. 同时, 每次参与计算的矩阵元素增加, 进而达到有效利用 GPU 并行计算能力的目的. 为充分利用 GPU 并行计算能力, 可以在程序中显式使用向量数据类型. 相对标量数据类型, 向量数据类型可由 2–4 个 32 位的标量数据类型组合而成, 其大小由向量数据类型的后缀数字指示. 例如, 例如, float4 向量数据类型由 4 个 float 数据类型构成, 分别用 x、y、z 和 w 进行引用. 使用 float4 向量数据类型, 可以在一次存储访问请求中取入 4 个 float 类型的标量数据. 与标量数据类型相比, 使用向量数据类型后的存储访问次数有所减少, 较大限度地利用了存储带宽, 如图 4 所示.

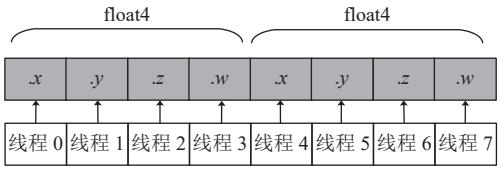

图 4 float4 向量数据类型

2.3.2 相似度计算加速

在事件结构相似度和情景相似度计算中, 由于不存在数据依赖关系, 因此可以安排每个 GPU 线程负责目标事件与网格事件库中的一个历史事件进行相似度计算. 为计算情景相似度, 本文实现了基于 GPU 的 Word-Similarity 计算过程[18]. 在 WordSimilarity 并行化过程中, 关系义原描述中存在大量的分支结构, 在 GPU 上并行化会严重影响执行性能. 为此, 对于一些以线程作为控制变量的条件分支, 在控制变量取值有限的情况下, 本文通过引入数组实现分支路径的间接索引, 如代码 1 和代码 2 所示. 引入的数组可以通过线程的进行访问, 该数组返回的值即是相应的分支路径. 也就是说, 每个线程根据其访问数组来决定其所要执行的分支路径, 这种间接索引的方式能够完全消除条件分支.

代码 1. 分支重构前代码

```
if(tid.x==0)
   if(tid.y==0) statements with func(m,a,b);
   else statements with func(m,a);
else if(tid.x==1)
   if(tid.y==0) statements with func(m,a,b);
   else statements with func(m,a,b);
else if(tid.x==2)
   if(tid.y==0) statements with func(m,a,b);
   else statements with func(m,a,b);
else if(tid.x==2)
   if(tid.y==0) statements with func(m,a,b);
   else statements with func(m,a,b);
```

代码 2. 分支重构后代码

```
index=m+2*tid.x;
array[2][4] = {(1,2,2,1),(2,1,1,2)};
a[array[tid.y][tid.x]]=func(b[index],a[tid.y]);
```

3 实验分析

为验证本文提出的网格事件相似度并行计算的性能, 选取安徽省芜湖市 2016 年 1 月 1 日–2020 年 12 月 31 日期间 40 000 条社会化网格治理系统中的事件来构建数据集. 数据集中包含了事件属性、事件文本、事件情景描述、经过人工标注的相似事件等信息. 通过对 40 000 条事件文本进行统计分析, 长度均值为 632 个字, 且 95% 的事件情景描述的文本长度在 142 个字以内. 本文选取了其中 30 000 条事件作为训练集, 5 000 条事件作为验证集, 5 000 条事件作为测试集. 实验硬件环境为: 内存 DDR4 64 GB, 2.4 GHz Intel(R) Xeon(R) Silver 4210R CPU, NVIDIA GeForce RTX 3090 GPU; 软件环境为 Ubuntu 18.04, CUDA Toolkit 10.2.

3.1 事件相似度性能对比

本文采用精确率 (Precision)、召回率 (Recall) 和 $F1$ 值作为事件相似度计算性能的评价指标. 本文对 40 000 个网格事件采用第 2.2.2 节所述阈值计算方法, 得出事件相似度阈值为 0.91. 为了验证基于关键词生成的事件相似度性能, 通过实验与使用 TF-IDF、textRank 和 LDA (latent Dirichlet allocation) 对网格事件文本进行关键词提取并计算相似度的机器学习方法进行了对比, 对比结果如表 1 所示.

表 1 基线模型对比实验结果 (%)

计算方法	Precision	Recall	$F1$
TF-IDF	65.26	57.63	61.21
textRank	70.85	67.34	69.05
LDA	76.77	73.25	74.97
本文方法	**83.17**	**78.64**	**80.84**

通过对比可以发现, 本文基于深度学习的方法在性能上优于 TF-IDF 等基于机器学习的方法, 原因在于机器学习模型方法仅简单的对文本中的词向量进行加权平均, 没有使用文本更深层次的语义信息, 而深度学习方法可以获取更深层次的语义信息. 此外, 使用 attention 机制的 PGN 关键词生成模型更加关注那些对判断事件相似度因素贡献较大的文本特征. 通过消融实验将记忆网络从 PGN 中消除, 精确率指标为 82.61%. 可见, 记忆网络在精确率指标上能够提升 0.7% 的性能.

3.2 并行计算加速比分析

在 GPU 并行加速过程中, 实验采用每个线程块包含 512 个线程的设置. 从表 2 中看出, 当待比对的事件规模过小时 (5 000 件), GPU 的计算速度反倒比 CPU 慢, 一方面是因为 GPU 计算资源不能被充分利用, 另外一方面原因是 CPU 和 GPU 之间的数据传输开销及 kernel 启动开销抵消了 GPU 加速的性能增益. 当事件规模增加到 10 000 件时, GPU 并行加速的优势开始体

现. 当事件规模增加到 40 000 件时, 此时能够取得 4.04 倍的加速比. 从表 2 也可以看出, 随着事件规模的不断增大, 数据传输开销占比呈递减趋势. 可以预期的是, 随着事件规模的不断增加, 能够取得越来越高的加速比.

表 2 并行计算加速比

网格事件规模	加速比	数据传输开销占比 (%)
5 000	0.82	61.80
10 000	1.36	48.73
20 000	2.18	35.02
30 000	2.93	30.77
40 000	4.04	21.18

3.3 与 QRNN 和 SRU 网络的性能对比

准递归神经网络 (QRNN) 是一种交替卷积层的神经序列建模方法, 其网络结构由类似于 CNN 中的卷积层和池化层两类子部分组成[19]. QRNN 卷积层与 CNN 卷积层类似用于提取输入特征, 池化层可用于减少特征数目, 但不同的是 QRNN 采用了 fo-Pool (在动态平均池化的基础上增加了输出门) 结构进行池化. 简单循环单元 (SRU) 是一种轻循环单元[20], 其具体结构分为两个部分: 轻循环部分和高速网络 (highway network) 部分. 轻循环部分处理输入向量并计算包含序列信息的状态序列, 高速网络部分促进基于梯度的深度网络训练. 为验证 LSTM 网络应用在基于 PGN 的事件关键词生成的优越性, 本文采用 QRNN 网络和 SRU 网络对 40 000 个网格事件与 LSTM 网络进行了性能对比, 对比结果如表 3 所示.

表 3 LSTM、SRU 和 QRNN 性能对比

网络	层数	Precision (%)	Recall (%)	*F*1 (%)	加速比
LSTM	2	83.17	78.64	80.84	4.04
SRU	2	81.77	77.45	79.55	5.22
SRU	8	82.78	78.19	80.42	3.96
QRNN	2	80.96	76.81	78.83	4.98
QRNN	8	82.37	78.01	80.13	3.81

从表 3 可以看出: (1) QRNN 网络计算性能总体低于 SRU 网络; (2) 2 层 SRU 网络在计算速度上比 2 层 LSTM 网络要快, 但 2 层 SRU 网络在相似度计算精度上相对较弱; (3) 8 层 SRU 网络能达到和 2 层 LSTM 网络相近的相似度计算精度, 但 8 层 SRU 网络计算速度稍慢于 2 层 LSTM 网络. 可见, 虽然 SRU 网络在 LSTM 网络的计算方式上进行了优化, 但在本文任务上 SRU 网络的适应性弱于 LSTM 网络.

4 结论与展望

通为了满足大规模网格事件环境下实时计算事件相似度的需要, 本文提出了一种基于关键词生成的网格事件相似度并行计算方法. 该方法通过指针生成网络生成网格事件关键词, 基于关键词结构相似度和情境相似度计算事件相似度, 利用 GPU 对事件关键词生成过程中的 LSTM 网络和相似度计算过程进行加速. 实验结果表明: 相比 TF-IDF、textRank 和 LDA 方法, 本文方法在相似度计算性能上更好, 采用 GPU 进行并行计算最高获得了 4.04 倍的加速比. 本文下一步工作是在网格事件相似度的基础上结合案例推理技术实现社会突发事件的辅助决策.

参考文献

1 孔钦, 叶长青. 基于案例推理的故障诊断算法. 计算机系统应用, 2016, 25(1): 181–186.

2 Wang A, Gao XD. A variable scale case-based reasoning method for evidence location in digital forensics. Future Generation Computer Systems, 2021, 122: 209–219. [doi: 10.1016/j.future.2021.03.019]

3 Gao JQ, Xia YF, Yin RJ, *et al*. Adaptive diagonal sparse matrix-vector multiplication on GPU. Journal of Parallel and Distributed Computing, 2021, 157: 287–302. [doi: 10.1016/j.jpdc.2021.07.007]

4 Wang ZH, Wang D, Li Q. Keyword extraction from scientific research projects based on SRP-TF-IDF. Chinese Journal of Electronics, 2021, 30(4): 652–657. [doi: 10.1049/cje.2021.05.007]

5 See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Vancouver: Association for Computational Linguistics, 2017. 1073–1083.

6 李军, 乔立民, 王加强, 等. 政务大数据环境下基于案例推理的网格事件数据应用研究. 情报理论与实践, 2019, 42(2): 151–157.

7 杨峰, 张月琴, 姚乐野. 基于情景相似度的突发事件情报感知实现方法. 情报学报, 2019, 38(5): 525–533. [doi: 10.3772/j.issn.1000-0135.2019.05.008]

8 邱俊安, 邱奇志, 周三三, 等. 词语语义相似度在突发事件案例检索中的应用. 武汉理工大学学报 (信息与管理工程版), 2020, 42(3): 272–278.

9 汪成亮, 黄利莹, 赵凯. 多粒度时空事件序列相似度算法研究. 北京理工大学学报, 2021, 41(1): 102–111.
10 徐绪堪, 李一铭. 基于情景相似度的突发事件多粒度响应模型研究. 情报科学, 2021, 39(2): 18–23.
11 陈健瑶, 翟姗姗, 夏立新, 等. 融合句法特征和句法相似度的网络舆情突发事件识别方法研究. 图书情报工作, 2021, 65(9): 41–50.
12 陈默, 张景祥, 胡恩华, 等. 基于结构化分析和语义相似度的食品安全事件领域数据挖掘模型. 食品科学, 2021, 42(7): 35–44. [doi: 10.7506/spkx1002-6630-20200505-027]
13 宋英华, 余侃, 吕伟, 等. 城市公交车火灾事件案例相似度匹配研究. 中国安全科学学报, 2017, 27(4): 163–168.
14 李想, 王卫兵, 尚学达. 指针生成网络和覆盖损失优化的 Transformer 在生成式文本摘要领域的应用. 计算机应用, 2021, 41(6): 1647–1651.
15 牛长安, 葛季栋, 唐泽, 等. 基于指针生成网络的代码注释自动生成模型. 软件学报, 2021, 32(7): 2142–2165. [doi: 10.13328/j.cnki.jos.006270]
16 Borna K, Ghanbari R. Hierarchical LSTM network for text classification. SN Applied Sciences, 2019, 1(9): 1124. [doi: 10.1007/s42452-019-1165-1]
17 Moirangthem DS, Lee M. Abstractive summarization of long texts by representing multiple compositionalities with temporal hierarchical pointer generator network. Neural Networks, 2020, 124: 1–11. [doi: 10.1016/j.neunet.2019.12.022]
18 葛斌, 李芳芳, 郭丝路, 等. 基于知网的词汇语义相似度计算方法研究. 计算机应用研究, 2010, 27(9): 3329–3333. [doi: 10.3969/j.issn.1001-3695.2010.09.034]
19 Bradbury J, Merity S, Xiong CM, *et al*. Quasi-recurrent neural networks. Proceedings of the 5th International Conference on Learning Representations. Toulon: ICLR, 2017. 1–11.
20 Lei T, Zhang Y, Wang SI, *et al*. Simple recurrent units for highly parallelizable recurrence. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 4470–4481.