边缘计算的物联网深度学习及任务卸载调度策略

苟 英[1] 李冀明[2] 魏 星[3]

[1](重庆商务职业学院出版传媒系 重庆 401331)

[2](重庆工程学院软件学院 重庆 400056)

[3](重庆工程学院计算机学院 重庆 400056)

摘 要 为解决物联网深度学习模型的网络性能和隐私问题,提出一种边缘计算的物联网深度学习应用及任务卸载策略,以优化网络性能,保护数据上传中的用户隐私。深度学习的多层结构适用于边缘计算,边缘节点上传缩减的中间数据,因此减少了从物联网设备到云服务器的网络流量。考虑到边缘节点有限的服务能力,提出一种边缘计算环境中最大化任务数量的卸载调度策略,优化边缘计算的物联网深度应用性能。实验结果表明,该策略能够在边缘计算环境中执行多个深度学习任务,并且性能优于其他物联网深度学习优化解决方案。

关键词 边缘计算 物联网 深度学习 卸载调度策略 用户隐私

中图分类号 TP393 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.08.023

DEEP LEARNING OF IOT AND TASK UNLOADING SCHEDULING STRATEGY FOR EDGE COMPUTING

Gou Ying[1] Li Jiming[2] Wei Xing[3]

[1](*Department of Publishing and Media, Chongqing Business Vocational College, Chongqing 401331, China*)

[2](*School of Software, Chongqing Institute of Engineering, Chongqing 400056, China*)

[3](*School of Computer, Chongqing Institute of Engineering, Chongqing 400056, China*)

Abstract To solve the network performance and privacy problems of IoT in deep learning model, we proposed an edge computing deep learning application and task uninstallation strategy of the IoT to optimize network performance and protect user privacy in data upload. The multi-layer structure of deep learning is suitable for edge computing, and edge nodes upload reduced intermediate data, thus reducing network traffic from IoT devices to cloud servers. Considering the limited service capability of edge nodes, we proposed an unloading scheduling strategy to maximize the number of tasks in edge computing environment to optimize the deep application performance of edge computing in the IoT. The experimental results show that the strategy can perform multiple deep learning tasks in the edge computing environment, and its performance is superior to other deep learning optimization solutions of the IoT.

Keywords Edge computing IoT Deep learning Unloading scheduling policy User privacy

0 引 言

通过互连的对象和设备及其虚拟表示来扩展当前的互联网已成为一种趋势,物联网(Internet of things, IoT)的贡献在于通过事物之间的互连数量以及为了社会利益而将处理后的信息转化为知识而创造的信息价值的增加。在 IoT 中,较大的数据量和较复杂的数据内容使得传统机器学习技术对于 IoT 数据的处理存在一些缺点,如大量的特征提取等,而深度学习只需将数据直接传递到网络,避免了大量特征处理过程。另外,深度学习可以针对不同问题自动提取新特征,可以精确地学习多媒体信息的高级特征,提高处理多媒体信息的效率[4]。

收稿日期:2018-11-27。重庆科技厅科技攻关项目(cstc2016jcyjA0469)。**苟英**,副教授,主研领域:大数据,软件工程。**李冀明**,讲师。**魏星**,高工。

深度学习就是堆叠多个层,即下一层的输入是上一层的输出,这样能够对输入信息进行分级表达。深度学习网络与传统神经网络相同之处是:二者均采用分层结构,即输入-隐藏-输出网络结构,同一层不相邻节点没有连接,跨层之间也没有连接;不同之处是:传统神经网络是人工选择特征,并映射到值,训练较慢,较容易过拟合,参数不好调整,深度学习是网络选择特征,采用逐层训练机制,这就使得深度学习比神经网络更适合大量和复杂数据的处理。

深度学习作为一种大数据分析工具,已经成为视觉识别、自然语言处理和生物信息学等许多信息学领域的重要处理方法[2-4]。目前已经有深度学习在 IoT 的应用,如深度学习根据智能电表收集的数据对家庭用电量进行预测[5]、基于深度学习的物联网灌溉系统等[6]。由于高效研究复杂数据的实用性,深度学习将在未来的 IoT 服务中发挥非常重要的作用[7]。

深度学习比传统的机器学习方法花费更少的时间来推理信息[8],在 IoT 中执行深度学习提高了 IoT 的性能。文献[9]提出一种基于物联网深度学习的负载均衡方案,通过分析大量的用户数据和网络负载来测量网络负载和处理结构配置,并应用深度信任网络方法实现物联网中的高效负载平衡。文献[10]提出了一种新的可穿戴物联网设备的深度学习模型,提高了音频识别任务的准确性。

大多数现有的深度学习应用仍然需要云辅助,文献[11]中提出一个结合深度学习算法和 Apache Spark 进行物联网数据分析,推理阶段在移动设备上执行,而 Apache Spark 部署在云服务器中以支持数据培训,这种双层设计与边缘计算非常相似,这表明可以从云端卸载处理任务。

由于数据传输的网络性能有限,集中式云计算结构已经不能处理和分析从 IoT 设备收集的大量数据[12]。边缘计算能够将计算任务从集中式云卸载到 IoT 设备附近的边缘,因此通过预处理过程传输的数据会极大地减少,这使得边缘计算成为 IoT 服务的另一个重要技术[13-15]。深度学习模型中每个网络层都可以快速缩小中间数据大小,直到找到足够的特征,深度学习模型的多层次结构使其适用于边缘计算,因为可以在边缘上卸载部分学习层,然后将缩减的中间数据传输到集中式云服务器。深度学习在边缘计算中的另一个优势是中间数据传输中的隐私保护。

为了提高物联网深度学习模型的性能,本文将物联网的深度学习引入到边缘计算环境中,以提高学习性能并减少网络流量。制定了与不同的深度学习模型兼容的弹性模型,由于不同的深度学习模型的中间数据量和预处理开销不同,本文提出了一个调度问题,在边缘节点的有限网络带宽和服务能力下最大化深度学习任务的数量。为了在调度中保证每个物联网的深度学习服务的服务质量(QoS),设计离线和在线调度算法来解决这个问题。实验结果表明,本文解决方案优于其他优化方法。

1 用于边缘计算的深度学习

物联网设备会生成大量数据并将数据传输到云端以供进一步处理,这些数据包括多媒体信息,如视频、图像和声音,或结构化数据,如温度、振动和光通量信息。有许多成熟的技术用于处理结构化数据,然后自动控制物联网设备。传统的多媒体处理技术需要复杂的计算,不适用于物联网服务,由于深度学习技术提高了多媒体信息处理的效率,越来越多的作品开始将深度学习引入多媒体 IoT 服务。

视频传感是一项重要的物联网应用,它在物联网网络中集成了图像处理和计算机视觉,识别来自物联网设备记录的低质量视频数据的对象仍然是一项挑战。由于深度学习在视频识别中准确性较高,因此认为它是具有深度学习的典型物联网应用。

深度学习提高了物联网服务的多媒体处理效率,因为特征是由多层提取的,而不是传统的复杂预处理,然而,通信性能将是提高处理效率的瓶颈。所收集的多媒体数据大小远远大于传统的结构化数据大小,因此网络将收集到的数据从 IoT 设备传输到云服务的性能很难得到提高。

边缘计算是将收集的数据从物联网设备传输到云服务一种解决方案,物联网包含边缘层和云层,用于连接物联网设备和云服务。边缘层通常由物联网设备、物联网网关和局域网中的网络接入点组成;云层包括互联网连接和云服务器。边缘计算的处理在边缘层中执行,在边缘计算环境中,由于只有中间数据或结果需要从设备传输到云服务,因此减少了传输数据和减轻了网络压力。

边缘计算非常适合于中间数据小于输入数据的任务。因此,边缘计算对于深层学习任务是有效的,因为在深层学习网络层中,提取的特征大小被滤波器减少。图 1 给出了本文提出的物联网深度学习任务的边缘计算结构,该结构由云层与边缘层以及典型的边缘计算结构组成。

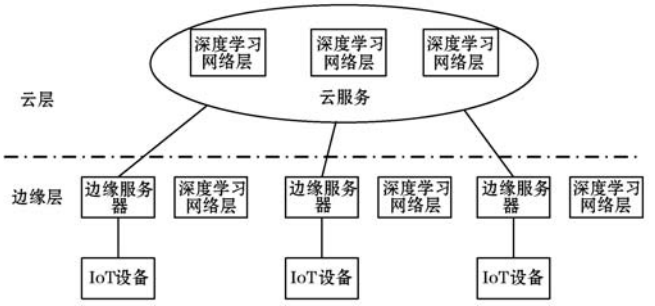

图 1　物联网深度学习的边缘计算结构

在边缘层中,边缘服务器部署在 IoT 网关中,用于处理收集的数据。首先在云服务器上训练深度学习网络,在训练之后,将学习网络分为两部分,一部分靠近输入数据的较低层,而另一部分靠近输出数据的较高层。将具有较低层的部分部署到边缘服务器中,将具有较高层的部分部署到云中用于卸载处理。因此,收集的数据被输入到边缘服务器中的第一层,边缘服务器从较低层加载中间数据,然后将数据作为较高层的输入数据传输到云服务器。

由较高层生成的中间数据小于由较低层生成的中间数据,将更多层部署到边缘服务器可以减少更多网络流量。但是,与云服务器相比,边缘服务器的服务器容量有限,在边缘服务器中处理无限任务是不可能的。深度学习网络中的每一层都会给服务器带来额外的计算开销,因此,只能将部分深度学习网络部署到边缘服务器中。同时,由于不同的深度学习网络和任务具有不同大小的中间数据和计算开销,在边缘计算结构中需要有效的调度来优化物联网的深度学习。针对这一问题,本文设计了一种有效的调度策略,并在下一节中进行了阐述。

物联网深度学习的边缘计算结构中深度学习网络使用的是深度自编码器,网络结构如图 2 所示。

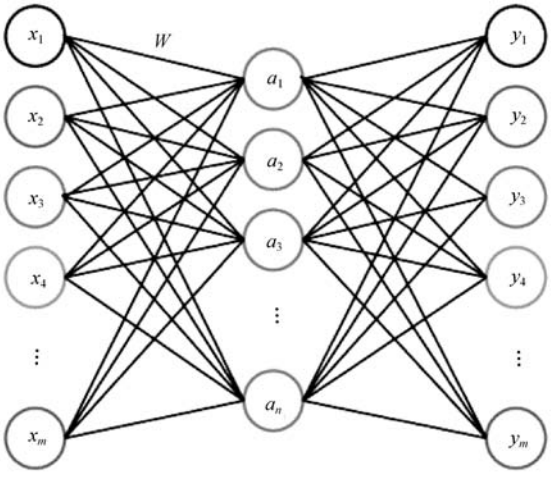

图 2　深度自编码器网络结构

对于输入向量 $\boldsymbol{x}$ 被编码为长度为低维特征向量 $\boldsymbol{a}$,然后被重构为近似于 $\boldsymbol{x}$ 的向量 $\boldsymbol{y}$,则在深度自编码器中,较低的维度 $\boldsymbol{a}$ 可以表示为:

$$\boldsymbol{a} = f\left(\sum \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right) \tag{1}$$

式中:$f(\cdot)$是输入映射函数,本文采用 sigmoid 函数,$\boldsymbol{W}$ 是权重向量,$\boldsymbol{b}$ 是偏置向量。

2　边缘计算中 IoT 深度学习的调度策略

首先阐述物联网深度学习的边缘计算结构中的调度问题,然后提出解决方案。在给定的边缘计算环境中,使用集合 E 来表示所有边缘服务器,e_i表示集合 E 中的边缘服务器。从边缘服务器 e_i到云服务器,使用值 c_i表示服务容量,b_i表示网络带宽。由于边缘服务器和云服务器之间存在一些交互流量,还添加了阈值 V 用以避免网络拥塞。因此,边缘服务器 e_i和云服务器之间的最大可用带宽由 $b_i \cdot V$ 表示。

移动节点 i 在每个时间段 t 中生成 $k_i(t)$比特的调度任务,对于 IoT 中 i 在 t 时的调度任务量定义为 $T_i(t)$,其在下一个时间段的更新规则可表示为:

$$T_i(t+1) = T_i(t) + k_i(t) - u_i(t) - v_i(t) \tag{2}$$

式中:$u_i(t)$表示本地任务执行量,$v_i(t)$表示卸载任务量。

t_j表示深度学习任务,任务 t_j'的深度学习网络层数为 N_j。假设减少的数据大小接近具有不同输入数据的每个任务的平均值。由第 k 层生成的中间数据大小($k \in [1, N_j]$)与总输入数据大小的平均比率由 r_{kj}表示。对于任务 t_j和边缘服务器 e_i,被分配的带宽由 b_{ij}表示。令 d_{ij}表示边缘服务器 e_i中任务 t_j的每时间单位的输入数据大小。如果任务 t_j的 k 个层放置在边缘服务器中,则边缘服务器 e_i中的任务 t_j的传送等待时间可以被表示为 $d_{ij} \cdot r_{kj}/b_{ij}$。为保证 QoS,传输延迟应小于 Q_j所表示的最大值。对于任务 t_j,第 k 层之后的输入数据单元的计算开销用 I_{kj}表示。因此,对于任务 t_j,计算机在边缘服务器 e_i中的开销是 $I_{kj} \cdot d_{ij}$。

计算中调度 IoT 深度学习网络层的问题:给定边缘计算结构,调度问题试图通过在 IOT 边缘服务器中部署深度学习层来分配边缘计算结构中的最大任务,使得所需的传输延迟每个任务都可以保证,如下式所示:

$$\max \sum_{i=1}^{|E|} \sum_{j=1}^{|T|} X_{ij} \tag{3}$$

$$\text{s.t.} \sum_{i=1}^{|E|} b_{ij} \leqslant b_i \cdot V$$

$$X_{ij} \cdot d_{ij} \cdot r_{kj}/b_{ij} \leqslant Q_j$$

$$\sum_{j=1}^{|T|} l_{kj} \cdot d_{ij} \cdot X_{ij} \leqslant c_i$$

如果任务 t_j 部署在边缘服务器 e_i 中，则 $X_{ij}=1$；否则，$X_{ij}=0$。

为了解决上述问题，本文提出了一种求解调度问题的离线算法和在线算法。离线调度算法首先找到使 $r_{kj}\cdot I_{kj}$ 的值最大的 k_j^m，和任务 t_j 的输入数据最大的边缘服务器 i_j^m，然后该算法按照最大输入数据大小的升序对所有任务进行排序。调度首先将具有最小输入数据的任务 t_j 部署到边缘服务器。该算法遍历所有边缘服务器，检查边缘服务器是否有足够的服务能力和网络带宽来部署任务 t_j。如果所有边缘服务器都足够服务容量和带宽，算法将任务 t_j 部署到所有边缘服务器中。如果边缘服务器没有足够的上传带宽或服务容量，该算法会改变 k 的值，并找出适合于所有边缘服务器中部署任务 t_j 的 k 值。如果边缘服务器即使在变化 k 后仍没有足够的服务容量或网络带宽，则调度算法将不会在边缘服务器中部署任务 t_j。

在最坏的情况下，离线算法的复杂度为 $O(|T|\cdot|E|^2\cdot K)$，其中 K 是每个任务的深度学习网络层的最大数量。由于任务数量远大于边缘服务器和深度学习网络层的数量，所提算法的复杂度为 $O(|T|)$，这对实际调度来说足够好，算法的效率近似为 $\frac{2}{V}$。

同时，设计一个在线调度算法，决定任务 t_j 到来时的部署。由于任务调度几乎没有关于特征任务的信息，因此部署决策是基于历史任务的。使用 $B^{\max}$ 和 $B^{\min}$ 分别表示任务的最大和最小期望带宽。因此，对于任务 t_j，首先计算 k_j^m 和 i_j^m，然后定义一个值：

$$\Phi(c_{ij}m)\leftarrow(B^{\min}\cdot e/B^{\max})^{c_{ij}m}\cdot(B^{\max}\cdot e) \quad (4)$$

式中：边缘服务器 $e_{ij}m$ 的剩余服务容量是 $c_{ij}m$。如果：

$$(b_{ij}m-d_{ij}m_j\cdot r_{ij}m_j/Q_j)\cdot(c_{ij}m-d_{ij}m_j\cdot I_{ij}m_j)\leqslant\Phi(c_{ij}m) \quad (5)$$

并且其他边缘服务器有足够的带宽和服务容量，则调度算法将任务 t_j 部署到边缘服务器。在线算法的近似比为：

$$\frac{1}{(\ln(B^{\max}/B^{\min})+1)\cdot V} \quad (6)$$

3 实验结果与分析

在本文实验中有两个实验环境，一个用于收集深度学习任务的数据，另一个用于仿真。使用配备为 Intel Core i7 7770 CPU 和 NVIDIA Geforce GTX 1080 显卡的工作站执行深度学习应用程序。实验定义了 10 个不同的深度自编码器网络，执行 10 个深度自编码器任务使用不同的网络并记录每个层中生成的操作和中间数据的数量。图 3 给出了深度网络的计算开销和减少数据大小比率指标，深度学习网络有五个层次和不同的神经元设置。

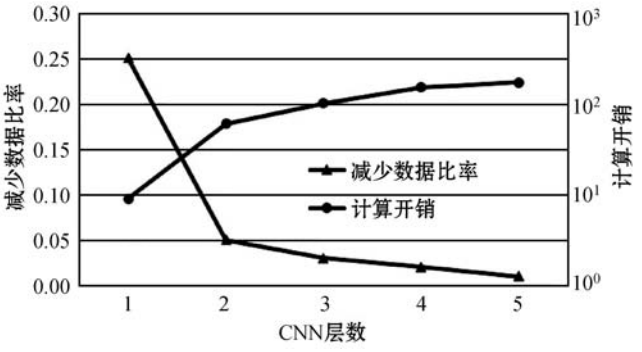

图 3 深度学习网络中的减少数据和计算开销

从图中可以看出，输入数据可以通过深度学习网络来减少，更多的中间数据可以通过较低层来减少，计算开销随着层数的增加而迅速增加。

使用 Python 2.7 和 networkx 开发模拟器，在模拟中，将深度学习任务的数量设置为 1 000。根据 NVIDIA Tegra K1，每台边缘服务器的服务能力设置为 290 Gflops，将网络中的边缘服务器数量从 20 增加到 90。每个任务的输入数据大小设置为从 100 KB 至 1 MB，所有深度网络的层数设置为 5 到 10。每个边缘服务器的带宽均匀分布在 10 Mbit/s 到 1 Mbit/s 之间，所需的延迟时间为 0.2 s。

将本文在线调度算法的性能与两种流行的在线调度算法：先入先出（FIFO）和低带宽优先部署（LBF）进行比较。向边缘网络输入 1 000 个任务的随机序列，这两种算法将任务部署到边缘服务器中，边缘服务器的数量设置为 50。图 4 给出了本文在线方法与其他两种方法的性能比较。

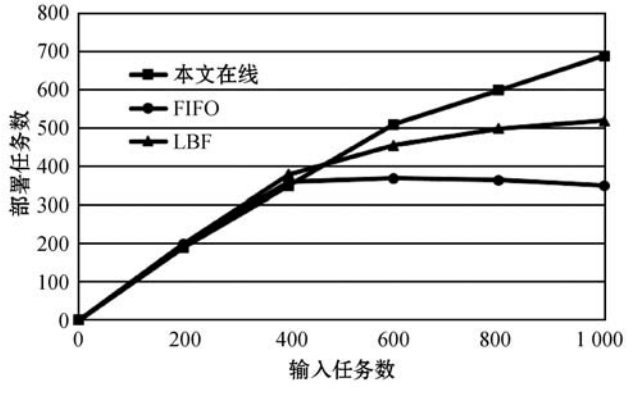

图 4 使用不同算法的部署任务的数量

如图 4 所示，FIFO 算法对每个任务进行部署，直到没有足够的能力和带宽，因此，在部署 360 个任务之后，FIFO 算法会弹出第一个部署的任务，用于附加以后的任务，部署的任务数量随着输入任务的增加不再增加，或者减少。当容量和带宽不足时，LBF 与 FIFO 类似，部署任务数不再随着输入任务数的增加而增加

了,当没有空间部署以后的任务时,LBF 将删除具有最大带宽要求的任务。本文的在线算法决定是否将以后的任务部署到边缘服务器中,当输入任务的数量接近 600 时,在线算法比 FIFO 和 LBF 能够部署更多的任务。因此,当容量和带宽不足时,本文在线卸载调度算法部署的任务数量会随着输入任务的增加,且优于 FIFO 和 LBF 算法。

4 结　语

本文将物联网的深度学习引入到边缘计算环境中,以优化网络性能并保护上传数据时的用户隐私。边缘计算结构使得边缘节点减少了上传数据的中间数据,减少了从物联网设备到云服务器的网络流量。针对边缘节点的有限服务能力,提出卸载调度算法来最大化边缘计算环境中的任务数量。实验结果表明,本文边缘计算的物联网深度学习应用及卸载调度算法可以在保证 QoS 要求的情况下增加边缘服务器中部署的任务数量。未来将利用本文算法在真实世界的边缘计算环境中部署深度学习应用。

参 考 文 献

[1] Li H, Ota K, Dong M. Learning IoT in edge: deep learning for the internet of things with edge computing[J]. IEEE Network, 2018, 32(1): 96 – 101.

[2] Gulshan V, Peng L, Coram M, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs[J]. Jama, 2016, 316(22): 2402 – 2410.

[3] Schmidhuber J. Deep learning in neural networks: An overview[J]. Neural networks, 2015, 61: 85 – 117.

[4] Ahmed E, Jones M, Marks T K. An improved deep learning architecture for person re-identification[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, MA, USA: IEEE, 2015: 3908 – 3916.

[5] Li L, Ota K, Dong M. When weather matters: IoT-based electrical load forecasting for smart grid[J]. IEEE Communications Magazine, 2017, 55(10): 46 – 51.

[6] Kwok J, Sun Y. A Smart IoT-Based Irrigation System with Automated Plant Recognition using Deep Learning[C]// Proceedings of the 10th International Conference on Computer Modeling and Simulation. New York, NY, USA: ACM, 2018: 87 – 91.

[7] Papernot N, McDaniel P, Jha S, et al. The limitations of deep learning in adversarial settings[C]// Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. Saarbrucken, Germany: IEEE, 2016: 372 – 387.

[8] Yao S, Zhao Y, Shao H, et al. ApDeepSense: Deep Learning Uncertainty Estimation without the Pain for IoT Applications[C]// IEEE, International Conference on Distributed Computing Systems. Vienna, Austria: IEEE Computer Society, 2018: 334 – 343.

[9] Kim H Y, Kim J M. A load balancing scheme based on deep-learning in IoT[J]. Cluster Computing, 2017, 20(1): 873 – 878.

[10] Bhattacharya S, Lane N D. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables[C]// ACM Conference on Embedded Network Sensor Systems Cd-Rom. New York, NY, USA: ACM, 2016: 176 – 189.

[11] Alsheikh M A, Niyato D, Lin S, et al. Mobile big data analytics using deep learning and apache spark[J]. IEEE Network, 2016, 30(3): 22 – 29.

[12] Liu J, Guo H, Nishiyama H, et al. New perspectives on future smart FiWi networks: scalability, reliability, and energy efficiency[J]. IEEE Communications Surveys & Tutorials, 2017, 18(2): 1045 – 1072.

[13] Shi W, Cao J, Zhang Q, et al. Edge computing: Vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637 – 646.

[14] Singh D, Tripathi G, Alberti A M, et al. Semantic edge computing and IoT architecture for military health services in battlefield[C]// 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC). Las Vegas, NV, USA: IEEE, 2017: 185 – 190.

[15] Sun X, Ansari N. EdgeIoT: Mobile edge computing for the Internet of Things[J]. IEEE Communications Magazine, 2016, 54(12): 22 – 29.

(上接第 64 页)

[8] Berger A, Caruana R, Cohn D, et al. Bridging the lexical chasm: statistical approaches to answer-finding[C]//International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2000:192 – 199.

[9] Jijkoun V, Rijke M D. Retrieving answers from frequently asked questions pages on the web[C]//ACM International Conference on Information and Knowledge Managementsigir Conference,2005:76 – 83.

[10] Riezler S, Vasserman A, Tsochantaridis I, et al. Statistical Machine Translation for Query Expansion in Answer Retrieval[C]//Meeting of the association for computational linguistics, 2007: 464 – 471.

[11] Baeza-Yates R A, Ribeiro-Neto B A. Modern Information Retrieval the Concepts and Technology Behind Search[M]. China Machine Press, 2011.

[12] Song W, Feng M, Gu N, et al. Question Similarity Calculation for FAQ Answering[C]//International Conference on Semantics, Knowledge and Grid. 2007:298 – 301.