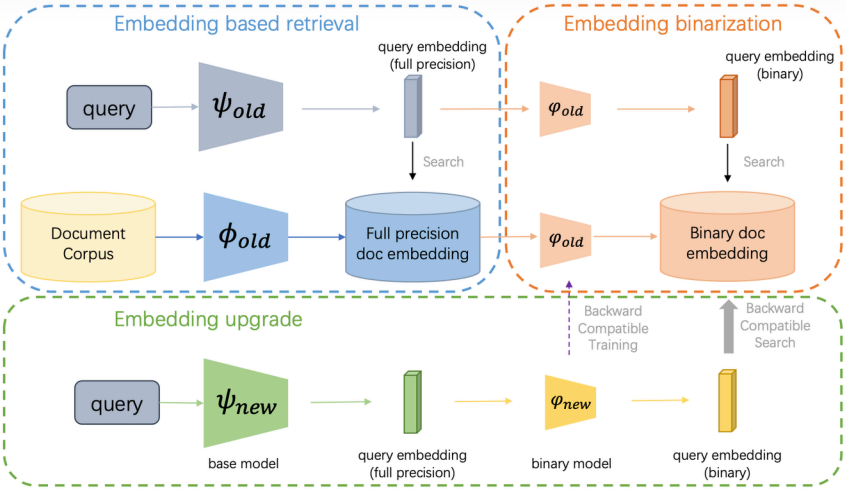
知乎




# 腾讯2023：二值化向量检索，揭示高效检索新途径

SmartMindAI
专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我

已关注

10 人赞同了该文章

## 原文：《Binary Embedding-based Retrieval at Tencent》

## Introduction

随着深度学习的发展，基于嵌入的检索在现实应 用中取得了很大进展。本文提出了一种 EBR 系统，通 过适当压缩的嵌入，可以与主流的 ANN 算法兼容，并可无缝集成到现有的 EBR 系统中。该系统由“召回-重 排”架构组成，使用 EBR 算法的召回模块的效率是整 个系统的瓶颈，因为它需要处理大量的文档。文档的 巨大规模和高并发查询给工业 EBR 系统带来了很大 的挑战，包括检索延迟、计算成本、存储消耗和嵌入 升级。尽管之前有尝试使用先进的 ANN 算法来提高 EBR 系统的效率，但升级所有现有系统可能会带来很 高的开发成本。因此，关注 EBR 最基本的组件，即嵌 入，可能是更好的选择。

本文主要贡献如下。

- 我们提出了一种基于二进制嵌入的检索（BEBR）引擎，可以有效地在腾讯产品的数百亿篇文档中进行索引。所提出的方法可以配备各种ANN算法，并无缝地集成到现有系统中。
- BEBR通过定制的循环二值化和对称距离计算，在大大降低内存和磁盘消耗的同时，获得了优异的检索性能。
- BEBR开发了一种通用的训练范式，适用于所有模态，无需访问原始数据和主干网络，*即*，二进制嵌入是以任务无关的嵌入到嵌入方式进行有效训练的。
- BEBR支持嵌入模型的向后兼容升级，即新模型可以立即部署，无需刷新索引嵌入。我们首次研究了二进制嵌入上的兼容学习。

## Related work

### ANN Methods

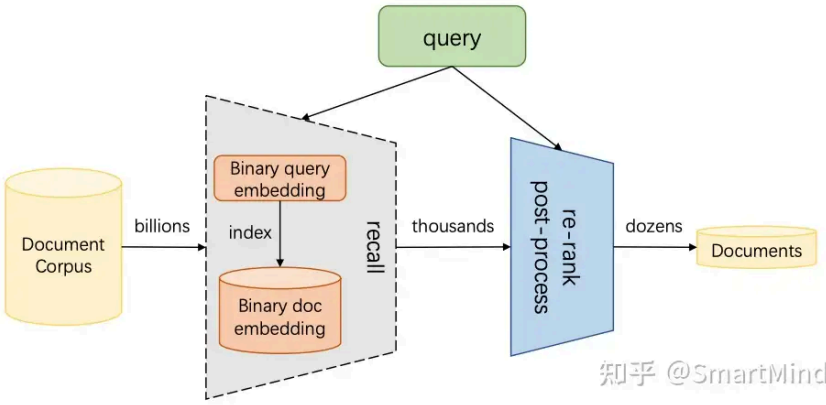
许多研究致力于开发高效的ANN算法。其中一些算法从数据集中构建图，以避免穷举搜索，图中每个顶点与数据点相关联。其他算法将嵌入编码为紧凑代码，以减少内存消耗并加快距离计算。

### Compatibility of Deep Neural Networks

兼容的表示学习旨在使模型之间的嵌入可比较。由于其降低嵌入升级计算成本的能力，它在工业和学术界吸引了越来越广泛的关注。具体来说，兼容性有两种类型：跨模型和后向兼容性。跨模型兼

知乎

示和回归。通过提出一个统一的表示学习框架，显著改善了跨模型兼容性性能。具体来说，他们设计了一个轻量级的残差瓶颈变换（RBT）模块，并使用分类损失、相似度损失和KL散度损失对其进行优化。 跨模型兼容性处理来自不同模型的嵌入，而后向兼容性则关注模型更新，其中新模型通过额外的兼容性约束进行训练。在某种意义上，它使新嵌入和旧嵌入之间无需任何额外的转换过程即可兼容。 是利用后向兼容性进行模型升级的第一项工作。它在训练新模型时引入了一个影响损失，使新嵌入和旧嵌入之间可以直接比较。在这个后向兼容性框架下，几项工作尝试通过利用热刷新后向兼容模型升级、不对称检索约束 、嵌入聚类对齐损失和神经架构搜索 来改善后向兼容性的性能。在本文中，我们采用后向兼容性训练来学习后向兼容的二进制嵌入。据我们所知，这是第一次将兼容性学习应用于二进制嵌入。



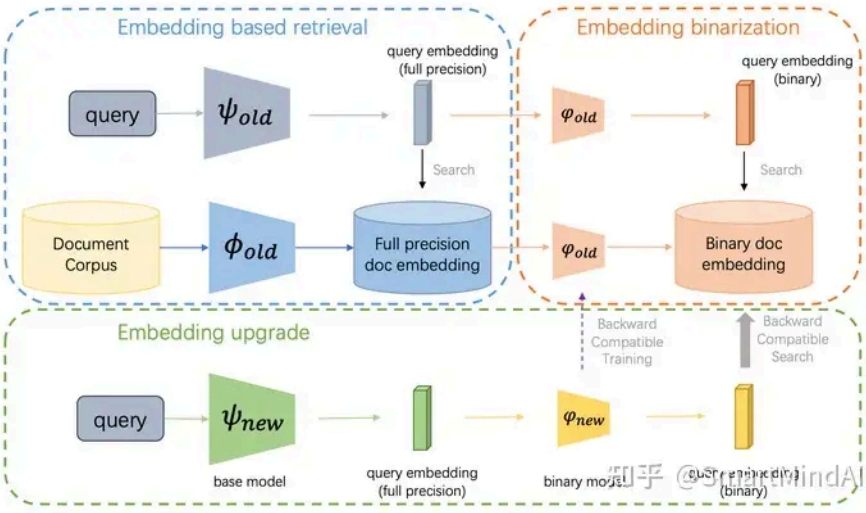

## Binary Embedding-based Retrieval

### Preliminary

给定一个查询$q$（通常是网络搜索⁺中的文本或版权检测中的视频），基于嵌入的检索（EBR）系统旨在根据其相似度对文档$\{d_0, d_1, \cdots, d_n\}$进行排名。 EBR中有两个关键因素，嵌入模型(s)和距离计算度量$\mathcal{D}(\cdot, \cdot)$。 对于全精度嵌入（浮点向量），广泛使用余弦相似度⁺作为$\mathcal{D}(\cdot, \cdot)$。形式上，某个查询和文档之间的相似度是

$$\mathcal{S}_{\mathrm{EBR}}(q, d_k) = \mathcal{D}\left(\psi(q), \phi(d_k)\right), \quad \forall k \in \{1, \cdots, n\},$$

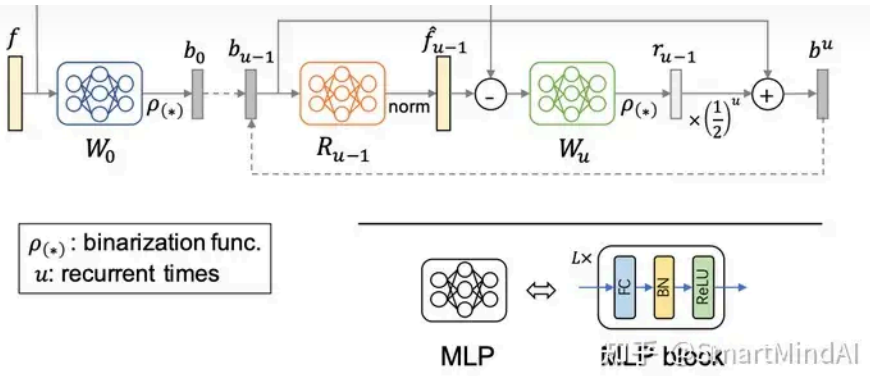
$\mathcal{S}_{\mathrm{BEBR}}(q, d_k) = \mathcal{D}\left(\phi \circ \varphi(q), \phi \circ \varphi(d_k)\right), \quad \forall k \in \{1, \cdots, n\}$, 其中$\varphi(\cdot)$是二值化⁺过程，一般通过参数网络实现。




### Recurrent Binarization

### Architecture

知乎




为了解决学习二值化的问题，直接的解决方案是采用哈希网络，最终得到一个二值化函数$\rho$，该函数在将浮点向量转换为由$-1$或$+1$组成的二值向量中起着重要的作用。在前向传播中，$\rho$被表述为

$$\rho(x) \equiv \operatorname{sign}(x) = \begin{cases} -1, & x \le 0 \\ 1, & x > 0 \end{cases}$$

具体来说，遵循使用残差操作来逐步缩小原始浮点向量和学习的二进制向量之间的差距的见解，我们引入了一个具有自定义循环的循环二值化模块，如图所示。 主要有三个组件，包括二值化块、重构块和残差块。二值化块的执行方式与传统的哈希网络几乎相同，其中二进制嵌入$b_0$从作为输入的浮点向量$f$编码

$$b_0 = \rho(W_0(f)) \in \{-1, 1\}^m,$$

其中$\rho$是二值化函数，$W_0$是多层感知机⁺（MLP），由线性、批量归一化⁺和ReLU层组成。 然后将编码的二进制嵌入$b_0$重构回浮点向量，如$\hat{f}_0 = \|R_0(b_0)\|$， 其中$R_0$也是多层感知机（MLP）。 因此，原始$f$和重构$\hat{f}_0$之间的残差反映了二值化过程的表示损失，可以通过重复上述步骤对残差部分进行二值化来进一步缩小。 残差二进制向量可以表示为

$$r_0 = \rho(W_1(f - \hat{f}_0)),$$

通过$b_1 = b_0 + \frac{1}{2} r_0$将其进一步添加到基本二进制向量$b_0$中。权重$\frac{1}{2}$的选择是为了简化相似性计算，只使用$xor$和$popcount$。 到现在为止，我们已经介绍了循环二值化的过程，当循环设置为$1$时，*即*，重复一次。所示，在以前的学习哈希方法中，骨干网络⁺$\phi$和二值化模块$\varphi$通常以端到端的方式进行联合优化。虽然可行，但鉴于用于准确表示学习的繁重骨干网络，训练效率不高，而且由于必须访问原始数据（如文本、图像）进行端到端训练，因此任务依赖性强，导致解决方案不够灵活，尤其是对于数据敏感的应用程序。 为了应对这一挑战，我们引入了一种通用的训练方案，只需要浮点向量作为输入，即使用现成的骨干网络$\phi$提取嵌入。因此，二值化模块$\varphi$以任务不可知和模态不可知的方式进行单独训练。这种嵌入到嵌入训练的目标函数可以表示为 $\arg\min_{\varphi} \mathcal{L}(\mathcal{F}; \varphi)$，其中$\mathcal{F}$表示用于训练的所有浮点向量的集合。 鉴于对比损失在表示学习研究中的巨大成功，我们采用NCE形式的对比目标来正则化二值化模块，

$$\mathcal{L}(\mathcal{F}; \varphi) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} -\log \frac{\exp\langle \varphi(f), \varphi(k_+) \rangle}{\sum_{k \in \mathcal{B}} \exp\langle \varphi(f), \varphi(k) \rangle},$$ 其中，$k$是与锚点$f$在同一批次$\mathcal{B}$内的浮点特征。$k_+$是由另一个增强视图的图像或来自网络的查询-文档对构建的正样本⁺。$\langle \cdot, \cdot \rangle$是循环二进制嵌入之间的余弦相似度。 除了通过手动注释或用户行为收集的正样本对之外，我们还采用硬负采样挖掘来进一步提高所学习二进制嵌入的辨别力。 硬负采样挖掘已被证明是提高分类任务准确性的一种有用技术。在深度学习社区中。最近关于语义检索⁺的工作 也成功应用了这种技术来提高检索准确性。有在线和离线硬负采样挖掘方法来收集足够硬的负样本，并提高模型识别相似但不相关查询-文档对的能力。在线硬负采样挖掘是高效的，因为它在mini-batches内部实时进行。离线硬挖掘在每个训练周期之前进行，即使借助ANN算法也非常耗时。然而，离线挖掘被证明更有效，因为它可以在整个训练集中搜索并发现最困难的样本。如何使全局硬挖掘像离线方法一样，同时保持在线方法的效率，是一个具有挑战性但关键的问题。受到的启发，我们通过维护一个负样本嵌入的队列$\mathcal{Q} \in \mathbb{R}^{L \times m}$来解决这个问题。具体来说，我们使用固定长度（*即*，$L$）的队列（比小批量大16倍）扩展小批量，并动态挖掘队列中的硬样本。在每个训练步骤，将当前小批量的二元嵌入添加到队列中，如果达到最大容量，则移除队列中最旧的小批量。请注意，我们执行二元化模块的动量更新，以对队列中的嵌入进行编码，以保持不同批次之间的潜在一致性，遵循的实践。

大的$k$个负样本的操作。 一旦学习到$\varphi$，就可以以高效的嵌入到嵌入范式生成查询和文档的循环二进制嵌入。训练和部署过程都是任务无关的，因为只需要全精度嵌入作为输入，这使得所有模态和任务的通用嵌入二值化成为可能。

### Backward-compatible training

$$\mathcal{S}_{\text{BEBR-BC}}(q_{\text{new}}, d^+_{\text{old}}) \geq \mathcal{S}_{\text{BEBR}}(q_{\text{old}}, d^+_{\text{old}}),$$
$$\mathcal{S}_{\text{BEBR-BC}}(q_{\text{new}}, d^-_{\text{old}}) \leq \mathcal{S}_{\text{BEBR}}(q_{\text{old}}, d^-_{\text{old}}),$$

其中，$d^+$表示与用户查询$q$相关的文档，$d^-$表示与用户查询$q$不相关的文档。$\mathcal{S}_{\text{BEBR-BC}}(\cdot,\cdot)$计算查询的新二进制嵌入与文档的旧二进制嵌入之间的相似度，具体公式为：

$$\mathcal{S}_{\text{BEBR-BC}}(q, d_k) = \mathcal{D}\left(\tilde{\phi} \circ \varphi_{\text{new}}(q), \phi \circ \varphi_{\text{old}}(d_k)\right) \quad \forall k \in \{1, \cdots, n\},$$

其中$\varphi_{\text{new}}(\cdot)$表示循环二值转换模块的新版本，而$\varphi_{\text{old}}(\cdot)$表示旧版本，$\tilde{\phi}$表示由特定应用确定的新的或相同的浮点主干模型。BEBR-BC代表向后兼容的BEBR系统。 训练目标可以表述为

$$\arg\min_{\varphi_{\text{new}}} \ \mathcal{L}(\mathcal{F}; \varphi_{\text{new}}) + \mathcal{L}_{\text{BC}}(\mathcal{F}; \varphi_{\text{new}}, \varphi_{\text{old}}),$$

其中$\mathcal{L}$与式相同，$\mathcal{L}_{\text{BC}}$也是NCE损失的形式，但跨越新旧模型，即

$$\mathcal{L}_{\text{BC}}(\mathcal{F};\ \varphi_{\text{new}}, \varphi_{\text{old}}) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} -\log \frac{\exp\langle \varphi_{\text{new}}(\tilde{f}), \varphi_{\text{old}}(k_+)\rangle}{\sum_{k \in \mathcal{B}} \exp\langle \varphi_{\text{new}}(\tilde{f}), \varphi_{\text{old}}(k)\rangle}.$$

$\tilde{f}$由$\tilde{\phi}(\cdot)$编码。$\varphi_{\text{new}}$在其他参数模块固定的情况下进行单独优化。$\mathcal{L}$保持新二值化模型的自辨别性，而$\mathcal{L}_{\text{BC}}$则规范化了跨模型的兼容性。基于队列的硬挖掘也应用于$\mathcal{L}_{\text{BC}}$。

### Deployment

### Dot product of recurrent binary embeddings

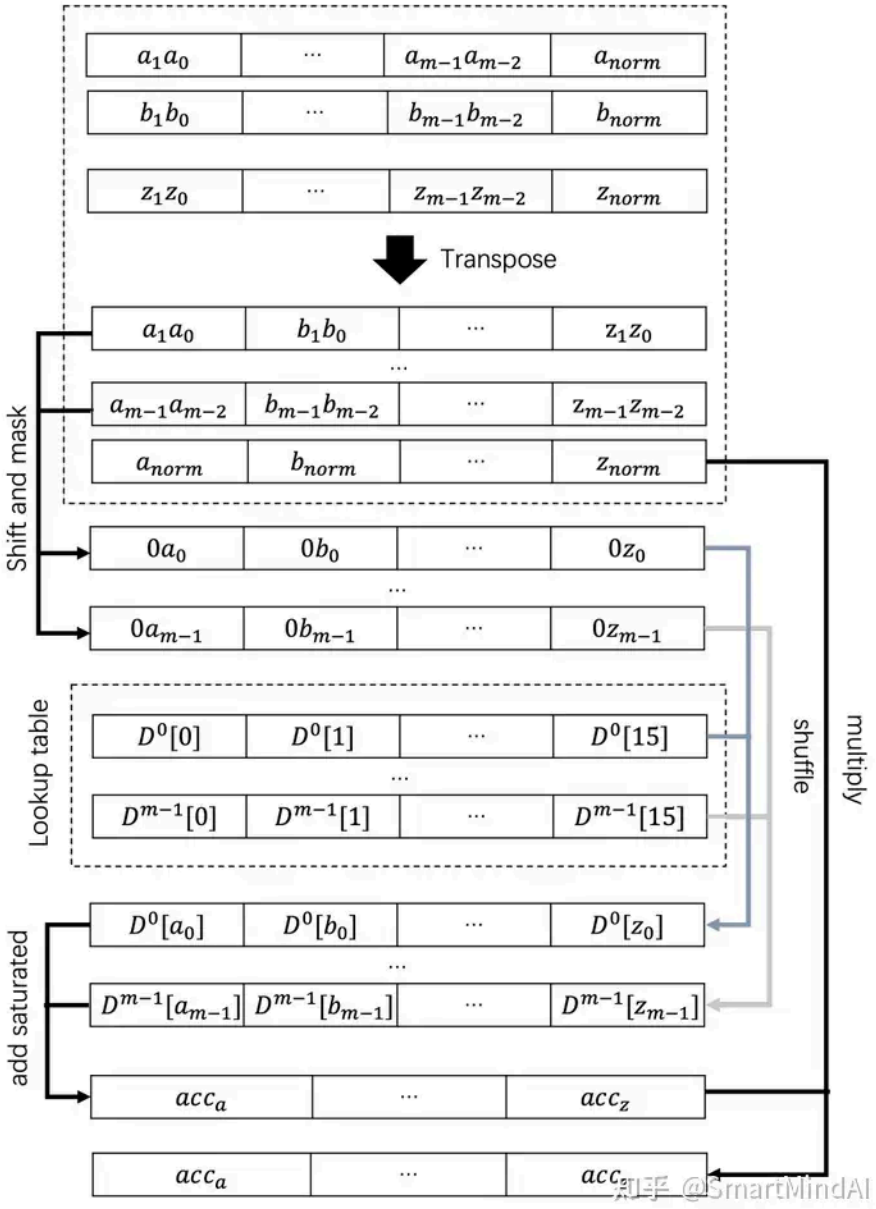
$$\mathcal{D}(b_u^q, b_u^d) \propto \frac{1}{||b^d||}(b_0^q \cdot b_0^d + \sum_{j=0}^{u-1}\sum_{i=0}^{u-1}(\frac{1}{2})^{j+i+2} r_j^q \cdot r_i^d + \sum_{j=0}^{u-1}(\frac{1}{2})^{j+1} b_0^q \cdot r_j^d + \sum_{i=0}^{u-1}(\frac{1}{2})^{i+1} b_0^d \cdot r_i^q)$$

$x \cdot y = (popc(x \wedge y) >> 1) + m$ 尽管按位运算在人口计数方面速度很快，但随着$u$的增加，计算复杂度迅速增加。因此，它依赖于GPU来提供高性能，并开发了一种优化的k-NN选择算法[+]。

### Symmetric distance calculation (SDC)

不幸的是，GPU启用的NN搜索算法限制了其在实际情况中的有用性和适用性。在本文中，我们开发了一种适用于大多数场景的围绕CPU平台的对称距离计算（SDC）的循环二进制嵌入。具体而言，SDC允许计算未压缩的循环二进制特征之间的距离。它依赖于SIMD寄存器内洗牌操作来提供高性能计算过程，可以与倒排索引[+]结合使用。为简单起见，我们在以下内容中解释SDC使用128位SIMD。，SIMD寄存器和寄存器内洗牌用于存储查找表[+]和执行查找。然而，这些方法在计算过程中使用子量化器来获得不同的质心而无需归一化。因此，需要进行算法更改以获得固定的质心和嵌入的幅度以进行归一化。更具体地说，SDC依赖于4位代码作为基本单位，并使用8位整数来存储查找表。所得查找表包括16个8位整数（128位）。一旦查找表存储在SIMD寄存器中，寄存器内洗牌可以在1个周期内执行16次查找，从而实现巨大的性能提升。 **内存布局。**通过设置$u \in 2, 4$，我们首先生成循环二进制向量并组织具有标准内存布局的特征倒排列表。如图上部所示。$a_i$是4位代码，$a_{norm}$是附加在向量末尾的量化幅度值。值得注意的是，对于$u = 2$，$a_i$表示特征的两个相邻维度。为了有效地洗牌查找表，需要将倒排列表的标准内存布局转置，因为转置后的数据在内存中是连续的，并且可以在单次内存读取中加载SIMD寄存器。这个转换过程是在线下进行的，不

围是8位整数，因此可以在128位寄存器中直接重构距离表。当$u = 2$时，我们使用循环二进制向量的两个相邻维度来形成4位代码，可以通过分别添加两个2位的内积结果来计算距离。




**SIMD计算。**在准备好查找表和倒排列表后，我们逐个块地扫描每个倒排列表。我们描绘了这个过程。首先，将索引编码以每个单元格8位的形式打包成128位寄存器，然后使用移位和掩码解包子代码。对于每组子代码，通过使用洗牌和混合操作的查找实现来产生部分距离。这个过程重复$um/4$次，并通过饱和运算将每个部分距离相加得到距离。最后，通过将其幅度值除以距离来对每个距离进行归一化。实际上，由于乘法运算在SIMD中很快，因此我们将距离乘以幅度值的倒数。

### ANN systems

我们部署了一个基于ANN搜索算法的分布式搜索系统，所示。在运行时，嵌入学习模型动态生成查询嵌入。然后，代理模块将查询分发到叶模块，主要搜索过程在这里进行。每个叶模块配备了各种具有对称距离计算的ANN索引，因为我们的工作与ANN算法正交，兼容任何类型的索引。因此，我们可以根据不同的产品需求选择不同的算法。例如，倒排索引（IVF）有两层用于嵌入搜索，一层是通过$K$-$means$算法将嵌入向量量化为粗略集群的粗糙层，另一层是有效计算嵌入距离的精细层。两层都可以使用循环二进制嵌入进行对称距离计算。最后，通过选择合并过程，将所有叶子的结果用于生成前N个结果。

## Experiments

我们使用Adam优化器，初始学习率设为0.02。softmax的温度参数$\tau$设定为0.07。当梯度的范数超过5的阈值时，我们也采用梯度裁剪。在配有8块Nvidia V100 GPU的服务器上训练时，二进制表示学习的批次大小设为4096，兼容学习的批次大小设为128。二值化实验基于PyTorch框架进行。 我们用C++实现了256位的SDC，使用编译器内部函数访问SIMD指令。选定g++ 8.1版本并启用SSE、AVX、AVX2和AVX512。此外，我们使用Intel MKL 2018进行BLAS运算。我们在基于Skylake的服务器上进行实验，这些服务器是腾讯云*实例，围绕Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz构建。

### Datasets

我们在公开数据集和工业数据集上对所提出的BEBR进行评估。对于公开数据集，我们使用来自MS COCO标注数据集的图像和文本数据。对于工业数据集，我们使用从两个应用中收集的数据。一个是网络搜索，给定用户搜索查询，返回相关网页。另一个是视频版权检测，在给定的参考视频数据集中识别出重复、复制和/或稍微修改过的视频序列（查询）版本。 **离线数据集:**对于网络搜索，我们从*搜狗搜索引擎*中收集用户的查询和点击搜索日志。经过数据预处理*后，训练集包含4亿个样本，我们使用额外的300万个样本进行评估。对于视频版权检测，我们使用从视频序列中提取的800万张图像来训练模型，并手动标注3万个查询和60万个参考图像进行验证。此外，我们使用COCO标注数据集，其中包含约11万个训练图像和5千个验证图像。对于训练和验证集中的每个图像，提供了五个独立的人工生成的标注。 **在线数据集:**我们在上述两个应用的生产环境中部署了所提出的BEBR。网络搜索文档的大小约为60亿，覆盖了互联网上最活跃的网页。视频版权检测中从视频序列中提取的图像嵌入的大小约为100亿。

### Evaluation Metrics

**离线评估。**我们使用Recall\@k指标来评估所提出的基于二进制的嵌入检索方法*的离线性能。具体来说，给定一个查询$q$，其相关文档$\mathcal{D}^+ = \{d_1^+, \cdots, d_N^+\}$，以及模型返回的前k个候选文档作为检索集$\hat{\mathcal{D}} = \{d_1, \cdots, d_k\}$。在实践中，$k \gg N$。Recall\@k定义为：$\textbf{Recall@}k = \frac{|\mathcal{D}^+ \cap \hat{\mathcal{D}}|}{N}$
关于这个问题，我没有相关信息。您可以尝试问我其它问题，我会尽力为您解答。

### Offline Evaluation

**循环二进制嵌入的有效性。**我们在公共（学术）和私有（工业）基准上研究了循环二进制嵌入的有效性。如表所示，我们与基线哈希（每维1位）和oracle浮点（全精度嵌入，每维32位）进行了比较。 对于学术数据集，我们使用MS COCO标题数据集进行了图像到文本的检索实验。具体来说，我们使用CLIP的ResNet101模型为图像和文本数据生成浮点嵌入。大小为16384位的浮点嵌入被压缩成大小为1024位的循环二进制嵌入和哈希向量，实现了16倍的压缩比。如表所示，循环二进制嵌入优于哈希嵌入，并与浮点嵌入取得了相当的结果。 对于工业数据集，网络搜索和视频版权检测中的浮点嵌入向量大小分别为8192位和4096位。我们通过将它们压缩成大小分别为512位和256位的二进制嵌入，采用相同的16倍压缩比设置。结果如表所示，我们在网络搜索和视频版权检测应用中实现了与浮点嵌入相当的检索性能，分别超过了哈希嵌入2.4%和3.9%。

### Online A/B Test

我们将基于二进制嵌入的检索系统*部署到腾讯的网页搜索和视频版权检测应用中，并与使用全精度嵌入进行检索的强大基线进行比较。请注意，我们仅在检索阶段用循环二进制嵌入替代全精度嵌入。随后的重新排序阶段对于这两种设置是相同的。在这里，我们想关注性能和效率的平衡，其中记录了资源使用情况。 实时实验在一周内对30%的服务流量进行。在系统级别上保持性能的同时，资源和效率的巨大优势。具体来说，BEBR节省了73.91%的内存使用量，并将检索的QPS提高了90%，而网页搜索应用的CTR和QRR分别略微下降了0.02%和0.07%。在视频版权检测中，内存使用量减少了89.65%，QPS提高了72%，而精度和检测率分别略微下降了0.13%和0.21%。检索效率和存储消耗的改进导致了整体成本的降低。部署BEBR后，网页搜索和视频版权检测中检索的总体成本分别降低了55%和31%。

## Conclusion

本文提出了基于二进制嵌入的检索（BEBR）方法，以提高检索效率，减少存储消耗，同时保持腾讯产品中的检索性能。具体来说,我们1）使用轻量级转换模型将全精度嵌入压缩为循环二进制嵌

知乎

计算，以形成有效的索引系统。BEBR已成功部署到腾讯的产品中，包括网页搜索（搜狗）、QQ和腾讯视频。我们希望我们的工作能够很好地激发社区将研究成果有效地转化为实际应用。

编辑于 2023-11-25 12:13 · IP 属地北京

搜索　推荐系统



**推荐阅读**


**向量数据库是如何检索的？基于 Feder 的 HNSW 可视化实现**

Zilliz

**大规模向量检索**

（一）什么是向量检索？ 我们知道，计算机只是一个电子设备的集合体，它没法像人一样感知这个世界。怎样使得计算机也能认识这个世界呢？计算机只认识数字，它只能通过数字来量化这个世界，...

yhmo

**【如果这篇说不清向量检索，那就来掐死我吧！】向量数据库...**

本文将介绍向量检索的几大经典算法： 图检索：NSW、HNSW、NSG；聚类中心：K-Means；乘积量化：PQ、IVFPQ；结合我看过的部分文章和博客，加入自己的思考，在我理解的范围内尽可能介...

CLAY

**如何玩转十亿向量检索（SIFT1B）——ZILLIZ**

文章目录 开始之前十亿向量
ANN_SIFT1B 数据集数据预
数据导入①数据预处理②数
数据检索①准确率查询②性
总结 开始之前请阅读以下文
了解 Milvus 的基本操作原理

格雷大大