Tencent 腾讯

CDG企业发展事业群

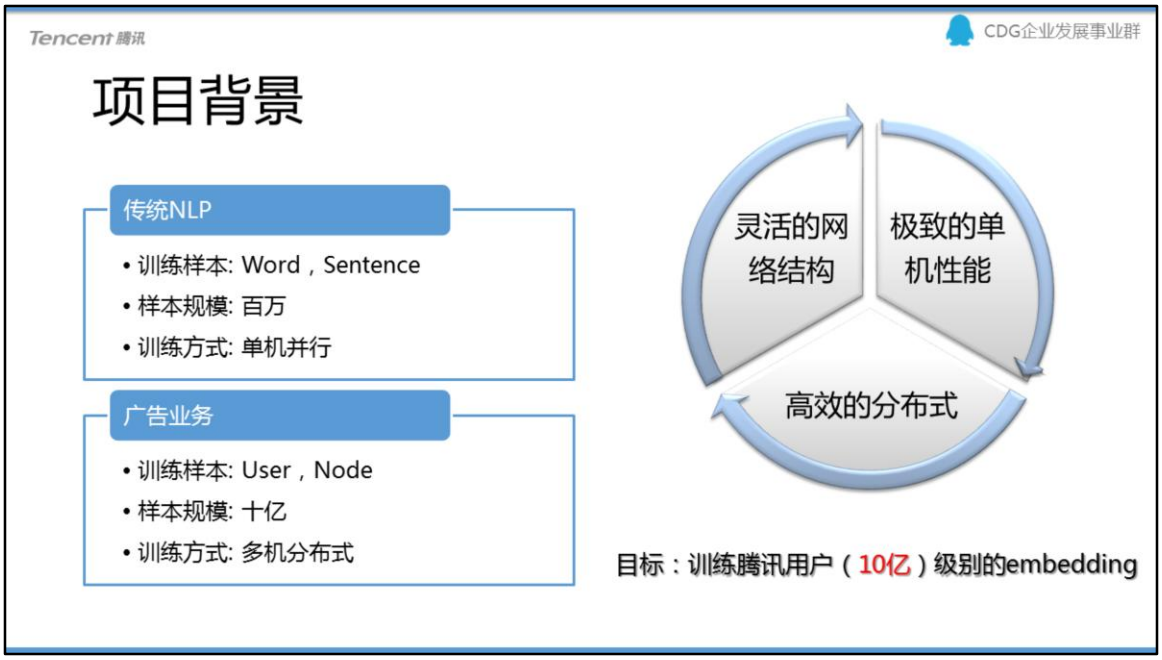
分布式AnyEmbedding系统设计与实现

Yuanhangzou, Kimmyzhang, Siweilai

2017年5月

目录

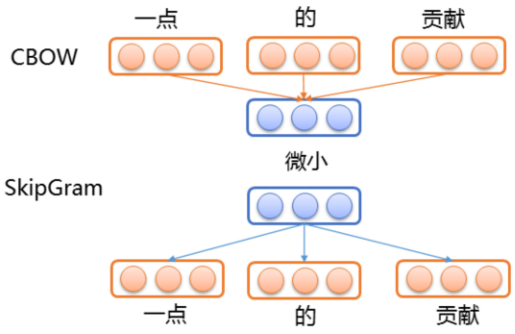
- 项目背景
- 灵活设计
- 高效实现
- 实验对比
- 应用前景



以腾讯用户规模为例，活跃QQ用户大约10亿，训练维度为100的embedding，使用float存储需要2 * 100 * 4 * 1000M=**800G**的内存

灵活设计

- CBOW模型
- SkipGram模型
- AnyEmbedding介绍



AnyEmbedding介绍

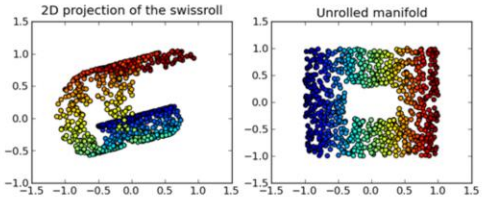
- Embedding是什么
- 为什么要做Embedding
- 为什么要做AnyEmbedding
- AnyEmbedding设计思想
- AnyEmbedding设计示例

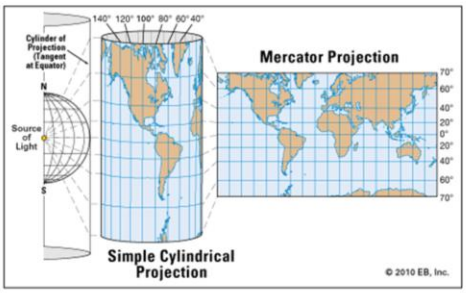
Tencent 腾讯

CDG企业发展事业群

Embedding是什么

- 嵌入：一个数学**结构**经映射包含到另一个结构中。 [Wiki]





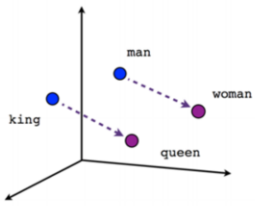
结构自定义
经纬度垂直

Tencent 腾讯

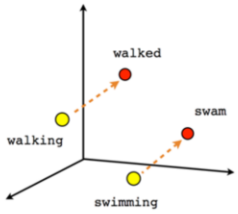
CDG企业发展事业群

Embedding是什么

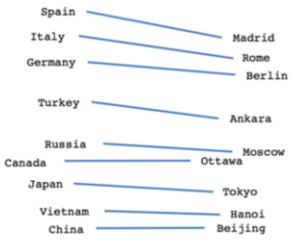
- 嵌入：一个数学**结构**经映射包含到另一个结构中 [Wiki]
- 词向量：上下文相似，则词义相似（分布假说） [Harris 1954]



Male-Female



Verb tense

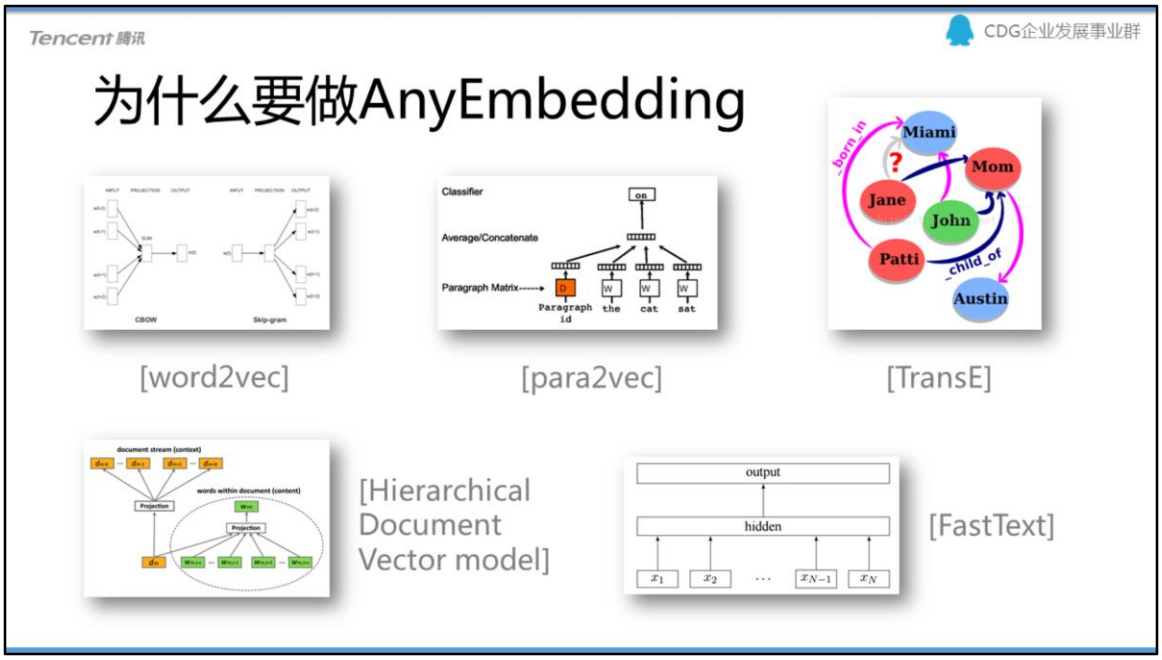


Country-Capital

上下文分布的结构

为什么要做Embedding

- 高维结构，低维表示
- 信息量大
- 存储少
- 检索方便，尤其对于图片 [Faiss]
- 直接对接神经网络



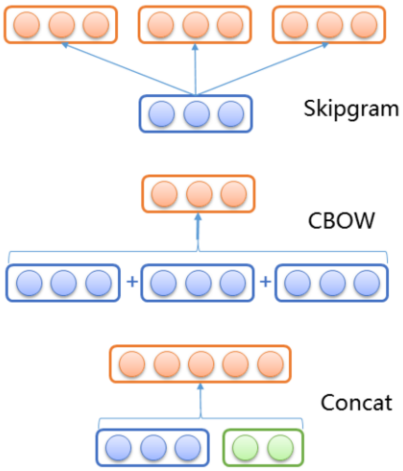
Embedding模型纷繁复杂

是时候做个AnyEmbedding一统天下了

TensorFlow、Caffe？

AnyEmbedding设计思想

- 通过“连接”构建模型
- 连接的形式
  - 一对一：一对多的简化
  - 一对多：Skipgram
  - 多对一：CBOW、 Concat
  - 多对多：看成多个多对一
- 只要能画出连接，就能做出模型

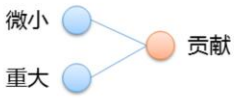


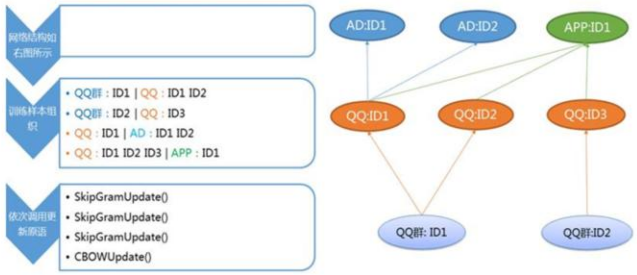
腾讯的数据

- 多源异构
- 连接一切

AnyEmbedding设计示例

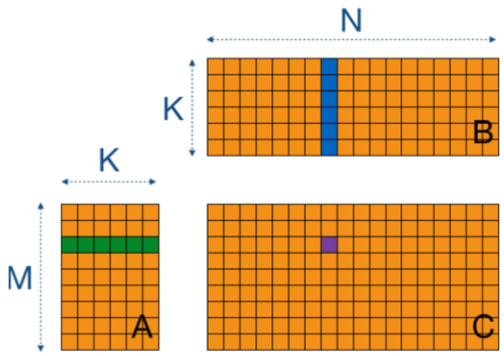
- 有关系就“连接”
- 明确任务目标
  - ➢有监督好于无监督
  - ➢主要连接作为目标
  - ➢其它连接作为正则
- 一阶关系二阶关系
  - ➢namespace





高效实现

- 向量化计算
- 共享负例
- 分布式训练架构

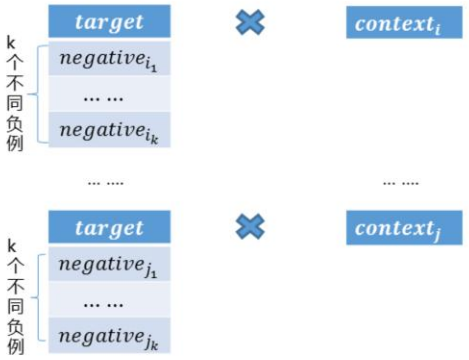


向量化计算

- 梯度和参数更新计算
  - Word2vec中梯度计算和参数更新用for循环实现的
  - 没有充分利用计算的潜力，比如对运算进行向量化
- BLAS介绍 [Wiki]
  - BLAS是基础线性代数操作的数值库，高性能计算领域广泛使用
  - BLAS有三级优化依次是向量-向量， 向量-矩阵， 矩阵-矩阵计算
  - 使用向量乘法代替for循环实现，效率有所提升但不显著

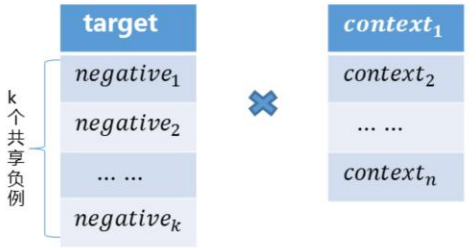
矩阵-向量计算

- 以SkipGram更新为例
  - 对于context_i，更新时的正样本是 $(context_i, target)$，负样本是 $(context_i, negative_{i_1})$ … $(context_i, negative_{i_k})$
  - 每个context正样本中target相同，负样本各不相同
  - 只能进行矩阵-向量计算，如右图

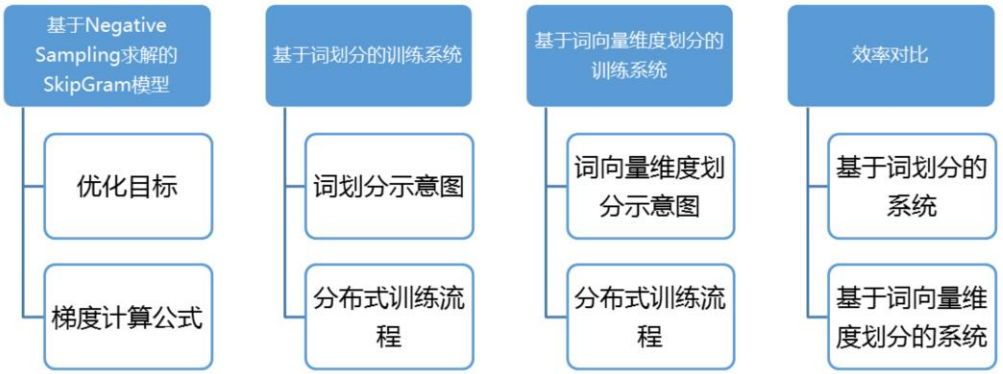


共享负例

- 如果每个context使用相同负例，即可进行矩阵-矩阵计算
  - 共享负例[Shihao Ji 2016]
- 矩阵-矩阵计算示例，如右图



分布式训练架构


基于Negative Sampling求解的SkipGram模型
优化目标
梯度计算公式
基于词划分的训练系统
词划分示意图
分布式训练流程
基于词向量维度划分的训练系统
词向量维度划分示意图
分布式训练流程
效率对比
基于词划分的系统
基于词向量维度划分的系统

基于Negative Sampling的SkipGram模型

- 优化目标公式

$$L = \sum_{i,j \in B_h} \sum_{\substack{k \neq j : |k-j| \leq b_{i,j} \\ w_{i,k} \in V}} [log \underbrace{\sigma(u(w_{i,j}) v^T(w_{i,k}))}_{prob\ of\ predicting\ w_{i,k}\ as\ pos} + \sum_{\hat{w} \in N_{i,j,k}} log(1 - \underbrace{\sigma(u(w_{i,j}) v^T(\hat{w}))}_{prob\ of\ predicting\ \hat{w}\ as\ neg})]$$

- 参数列表
  - V:词表，B_h: 一个batch的词集合， σ:sigmoid函数，$w_{i,j}$: B_h中的词， $b_{i,j}$上下文窗口大小
    $w_{i,k}$: $w_{i,j}$上下文中的词，$\hat{w}$: $w_{i,j}$负采样词，$N_{i,j,k}$:负采样词集合，u: input空间词向量，v: output空间词向量

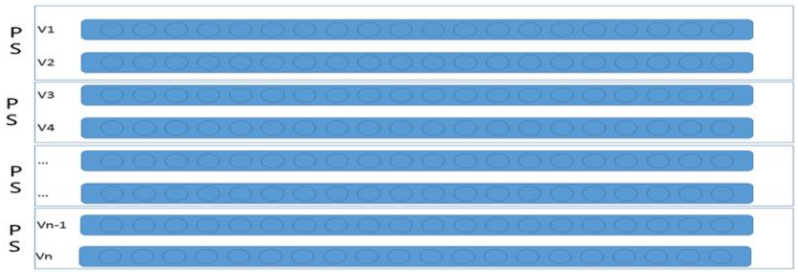
基于Negative Sampling的SkipGram模型

- 梯度计算公式
  - 以$u(w_{i_0,j_0})$为例

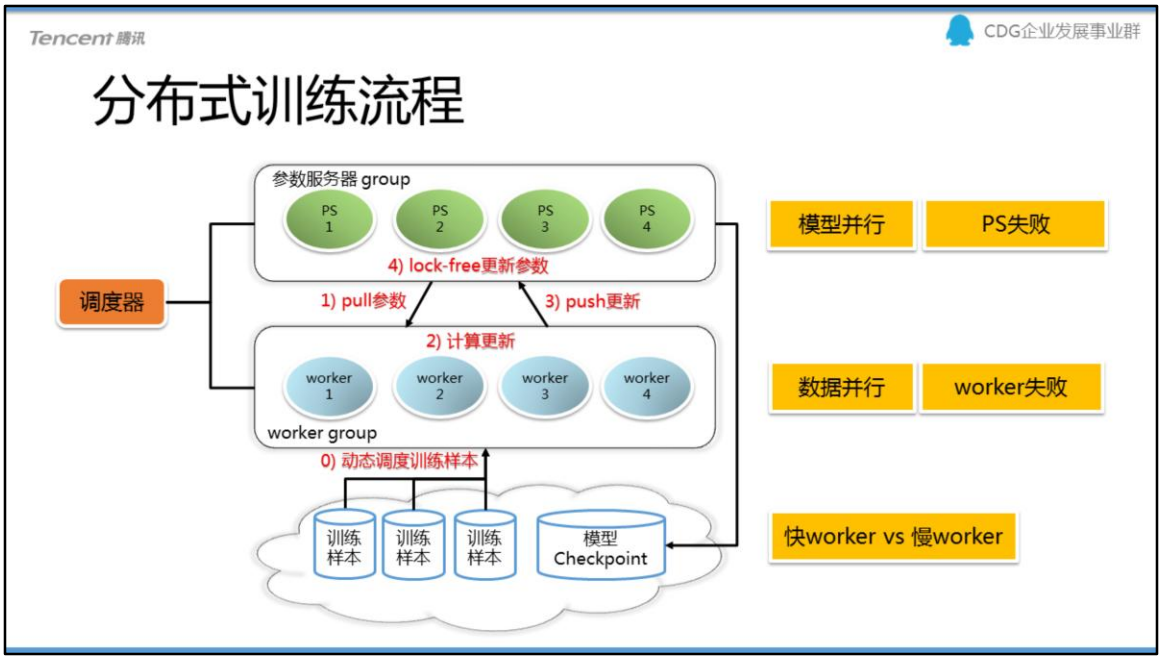
$$\nabla L|u(w_{io,jo}) = \sum_{\substack{(i,j)\in B_h: \\ w_{i,j}=w_{io,jo}}} \sum_{\substack{k\neq j, |k-j|\leq b_{i,j} \\ w_{i,k}\in V}} [\underbrace{(1-\sigma(\overbrace{u(w_{io,jo})v^T(w_{i,k})}^{dot\ \ product}))}_{linear\ \ combination} v(w_{i,k}) - \sum_{\hat{w}\in N_{i,j,k}} \underbrace{\sigma(\overbrace{u(w_{io,jo})v^T(\hat{w})}^{dot\ \ product})}_{linear\ \ combination} v(\hat{w})]$$

基于词划分的训练系统

- 每个server存储部分vocab的词向量整体



- worker计算梯度，server更新参数，通信时传递梯度向量



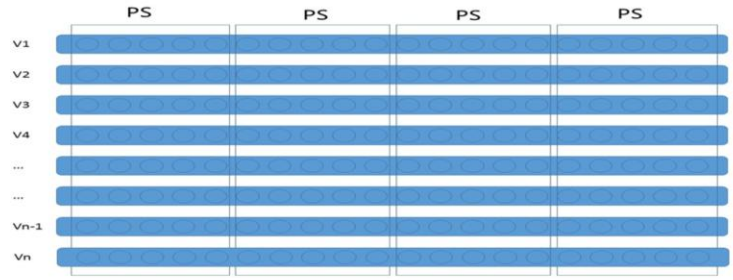
采用参数服务器构架: 参数服务器group，worker group，调度器

扩展性和容错性:
模型并行, 定时dump模型checkpoint, 供失败后恢复
数据并行

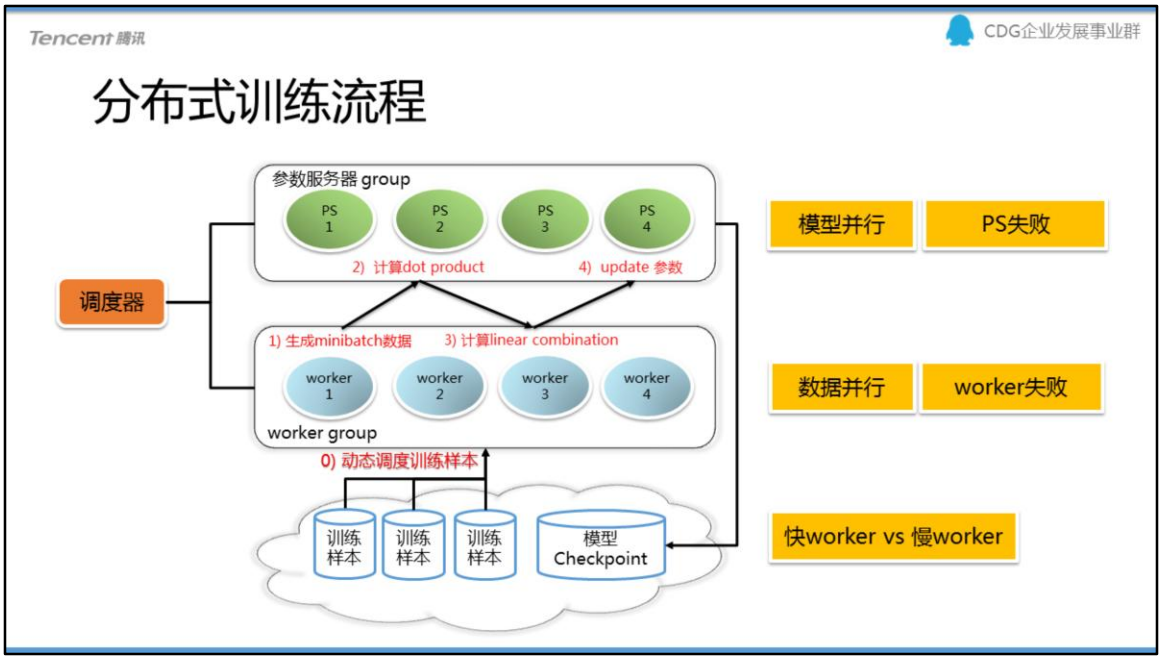
工作流程:
0-5
worker无状态 + 动态调度, 解决了worker失败和快慢问题

基于词向量维度划分的训练系统

- 每个Server存储全部词向量的部分维度



- 拆分梯度计算为向量乘法(server计算)和向量聚合(worker计算)，通信时传递向量乘法和聚合值，参数更新在Server计算 [Erik Ordentlich, 2016]



工作流程:

1) worker端，拉取minibatch训练数据，发送到所有server
2) server端，计算minibatch训练数据的embedding向量对应的dot product
3) worker端，拉取server端dot product结果，计算linear combination，发送到所有server
4) server端，利用worker端发送过来的linear combination结果，更新参数
5) 关于dot product和linear combination请参考第18页ppt介绍

效率对比

- 假设mini-batch词数量为b，上下文窗口为w，负例数量为n，server数量为s，词向量维度为d
- 基于词划分的系统
  - 通讯时传递训练数据对应的梯度，参数规模为 b * (w * n) * d
- 基于词向量划分的系统
  - 通讯时传递训练数据对应的向量乘积和向量聚合值，参数规模为b * (w * n) * s
- 基于词向量划分的训练系统通信效率提高了d / s
  - 通常d一般为100-300维，s为10-20台，加速可达10-30倍

实验对比

- 评测数据集
  - 使用了标准数据集，text8和1B-benchmark，后续更新更多数据集
- 效果对比
  - 对比在Word Similarity和Analogy中任务的效果
  - 后续更新实际业务中的效果
- 速度对比
  - 对比单机上AnyEmbedding和Word2vec的训练速度
  - 后续更新分布式对比结果

实验对比

- 结果对比

			Word Similarity		Word Analogy		Time	
Corpus	Vocabulary Size	Word Size	W2V	AED	W2V	AED	W2V	AED
17M-word(text8)	71291	16718843	0.562	0.559	0.113	0.113	757s	152s
1B-word benchmard	552403	796188544	0.514	0.531	0.143	0.14	12008s	3605s

- 评测任务和数据集
  - 效果上，在Word Similarity和Analogy任务上，评测结果基本一致
  - 时间上，AED相比于W2V加速明显，训练速度加速可达5倍
  - 共享负例做法可行，既可以加速训练又不影响效果

应用前景

User/Node Embedding

- 腾讯用户(10亿)规模的Embedding，用于相似&相关人群扩展

APP Embedding

- APP推荐

Query Embedding

- Query改写

… ….

单机和分布式系统已经在腾讯git.oa上面开源，欢迎使用

谢谢聆听