知乎搜索文本相关性与知识蒸馏

原创 申站 DataFunTalk 昨天

收录于话题

#原创精选 49 #知乎 1 #搜索算法 1

分享嘉宾：申站 知乎 算法工程师

编辑整理：许宴铭

出品平台：DataFunTalk

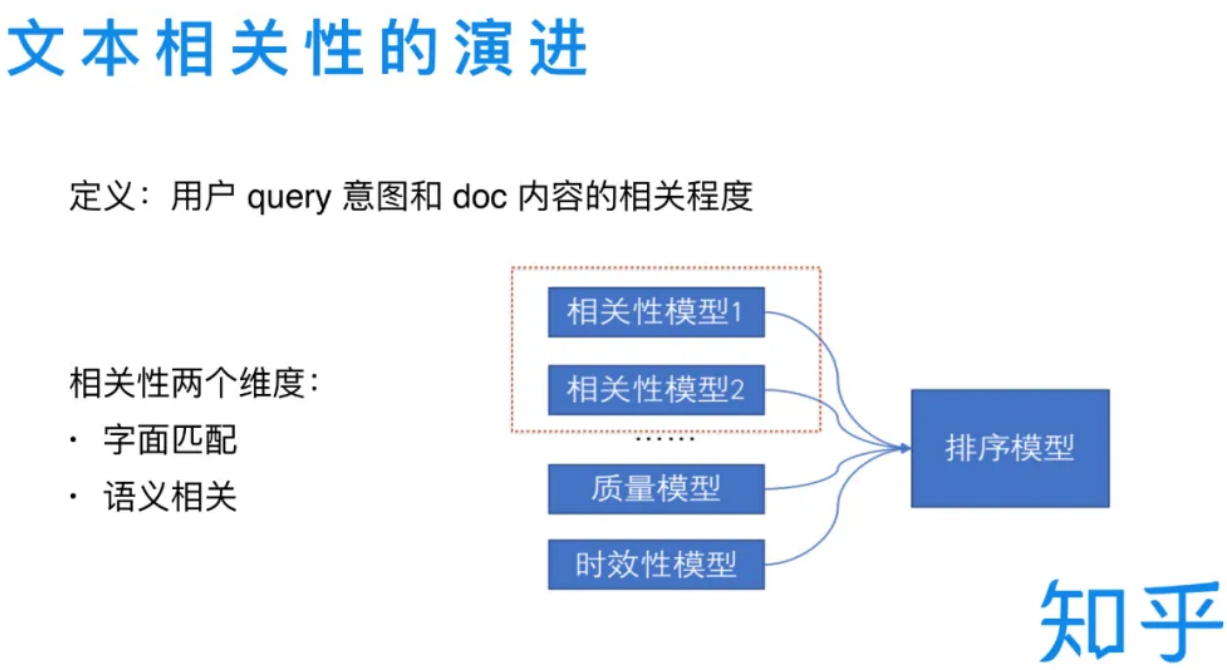
导读：大家好，我是申站，知乎搜索团队的算法工程师。今天给大家分享下知乎搜索中文本相关性和知识蒸馏的工作实践，主要内容包括：

- 知乎搜索文本相关性的演进
- BERT在知乎搜索的应用和问题
- 知识蒸馏及常见方案
- 知乎搜索在BERT蒸馏上的实践

01

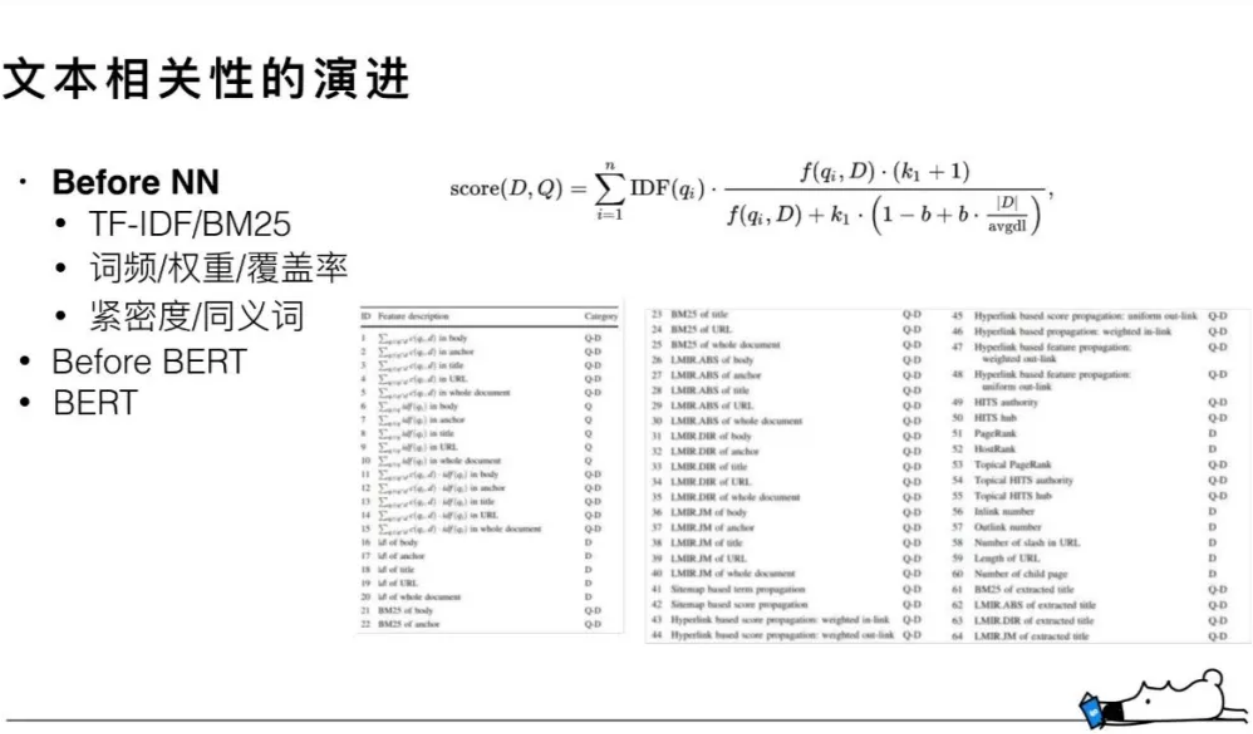
知乎搜索文本相关性的演进

1. 文本相关性的演进



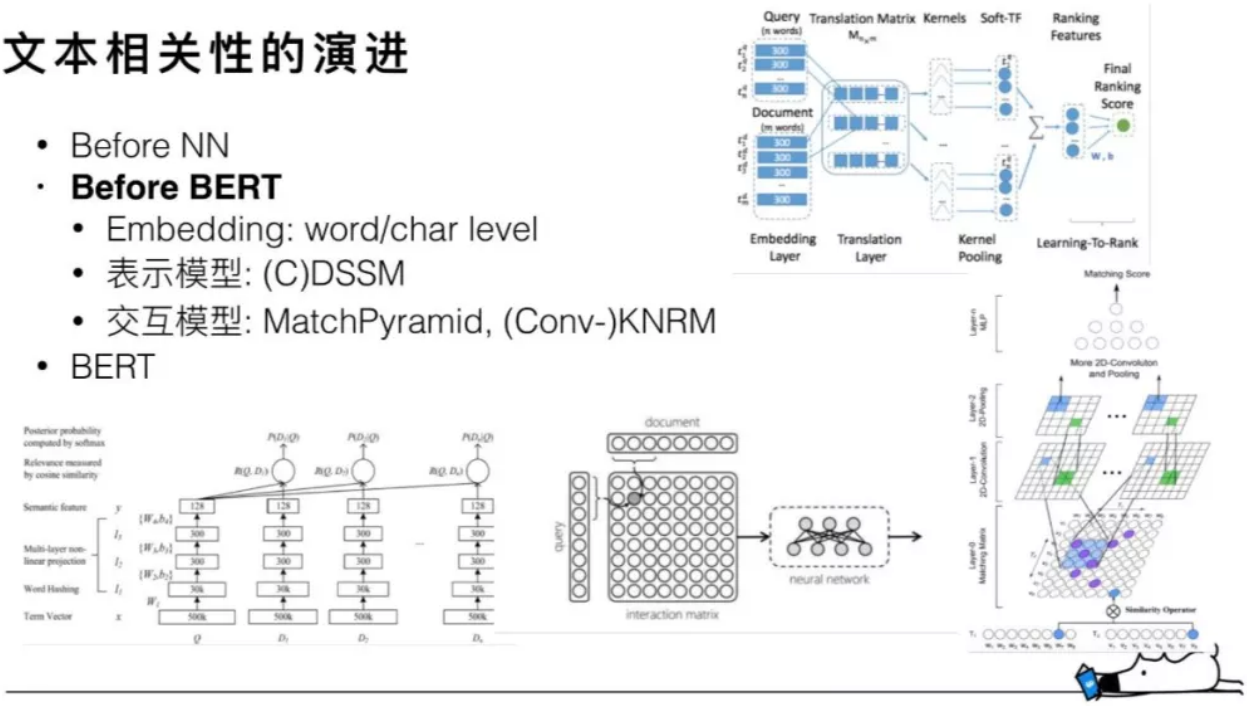
我们首先来介绍下知乎搜索中的文本相关性。在搜索场景中，文本相关性可以定义为用户搜索query的意图与召回 doc 内容的相关程度。我们需要通过不同模型来对这种相关程度进行建模。整体而言，文本的相关性一般可以分为两个维度，字面匹配和语义相关。知乎搜索中文本相关性模型的演进也是从这两个方面出发并有所侧重和发展。在知乎搜索的整个架构中，文本相关性模型主要定位于为二轮精排模型提供更高维/抽象的特征，同时也兼顾了一部分召回相关的工作。

2. Before NN



知乎搜索中的文本相关性整体演进可以分为三个阶段。在引入深度语义匹配模型前，知乎搜索的文本相关性主要是基于TF-IDF/BM25的词袋模型，下图右边是BM25的公式。词袋模型通常来说是一个系统的工程，除了需要人工设计公式外，在统计词的权重、词频的基础上，还需要覆盖率、扩展同义词，紧密度等各种模块的协同配合，才能达到一个较好的效果。知乎搜索相关性的一个比较早期的版本就是在这个基础上迭代的。右下部分为在基于词袋模型的基础上，可以参考使用的一些具体特征。

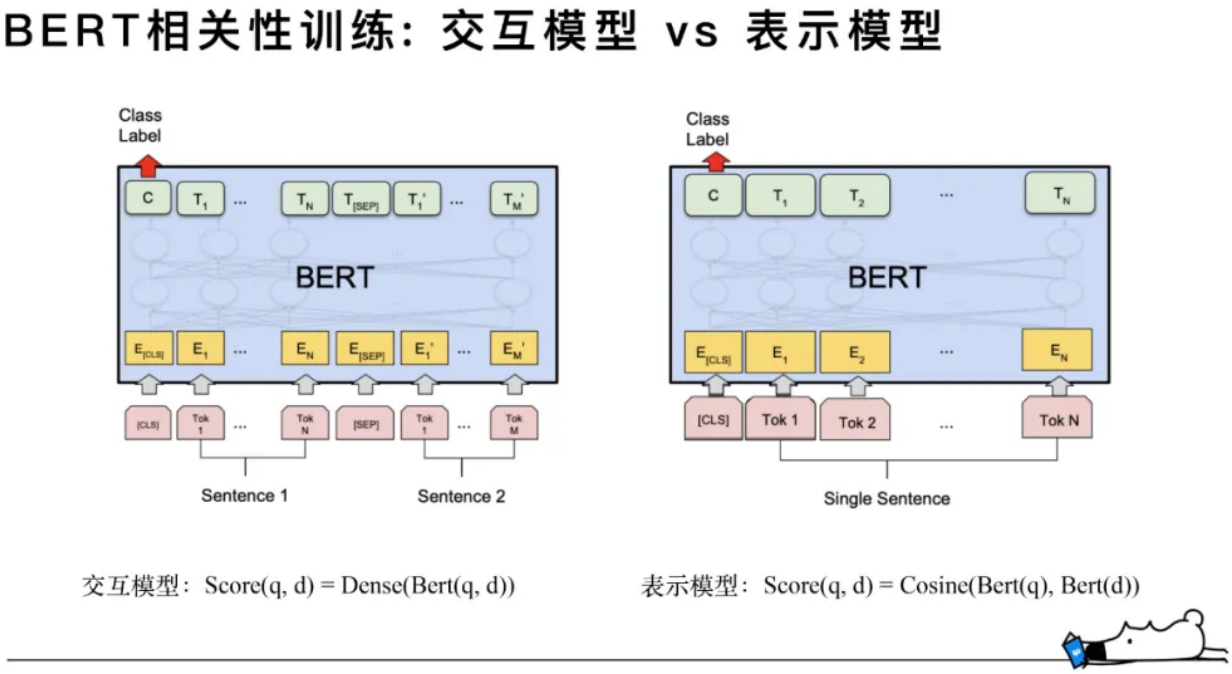
3. Before BERT



基于 BM25 的词袋模型不管如何设计，主要还是只解决文本相关性中的字面匹配这部分问题。第二阶段引入的深度语义匹配模型则聚焦于解决语义相关的问题，主要分为两部分：双塔表示模型和底层交互模型。微软的DSSM（左下）是双塔模型的典型代表。双塔模型通过两个不同的 encoder来分别获取query和doc的低维语义句向量表示，然后针对两个语义向量来设计相关性函数（比如cosine）。DSSM摆脱了词袋模型复杂的特征工程和子模块设计，但也存在固有的缺陷：query和doc的语义表示是通过两个完全独立的 encoder 来获取的，两个固定的向量无法动态的拟合doc在不同 query的不同表示。这个反应到最后的精度上，肯定会有部分的损失。

底层交互模型一定程度上解决了这个问题。这个交互主要体现在 query 和 doc term/char 交互矩阵（中）的设计上，交互矩阵使模型能够在靠近输入层就能获取 query 和 doc 的相关信息。在这个基础上，后续通过不同的神经网络设计来实现特征提取得到 query-doc pair 的整体表示，最后通过全连接层来计算最终相关性得分。Match-Pyramid（右下）、KNRM（右上）是交互模型中比较有代表性的设计，我们在这两个模型的基础上做了一些探索和改进，相比于传统的 BM25 词袋模型取得了很大的提升。

4. BERT



BERT模型得益于 transformer 结构拥有非常强大的文本表示能力。第三阶段我们引入了 BERT希望能够进一笔提高知乎搜索中文本相关性的表型。BERT 的应用也分为表示模型和交互模型。

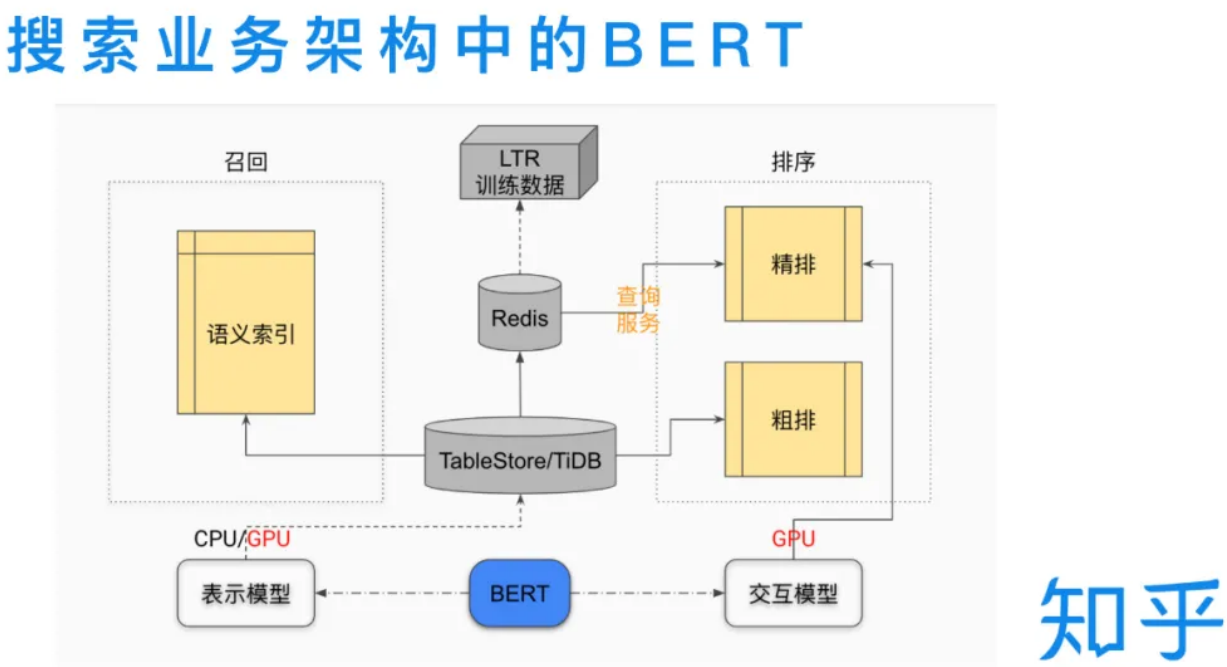
对于交互模型来说，如下左图，query和doc分别为sentence1和sentence2直接输入到BERT模型中，通过BERT做一个整体的encoder去得到sentence pair的向量表示，再通过全连接层得到相似性打分，因为每个doc都是依赖query的，每个query-doc pair都需要线上实时计算，对GPU机器资源的消耗非常大，对整体的排序服务性能有比较大的影响。

基于上述原因，我们也做了类似于DSSM形式的表示模型，将BERT作为encoder，训练数据的中的每个query和doc在输入层没有区分，都是做为不同的句子输入，得到每个句向量表示，之后再对两个表示向量做点乘，得到得到相关度打分。通过大量的实验，我们最终采用了 BERT 输出 token 序列向量的 average 作为句向量的表示。从交互模型到表示模型的妥协本质是空间换时间，因为doc是可以全量离线计算存储的，在线只需要实时计算比较短的 query ，然后doc直接通过查表，节省了大量的线上计算。相比于交互模型，精度有一部分损失。

02

BERT在知乎搜索的应用和问题

1. 搜索业务架构中的BERT



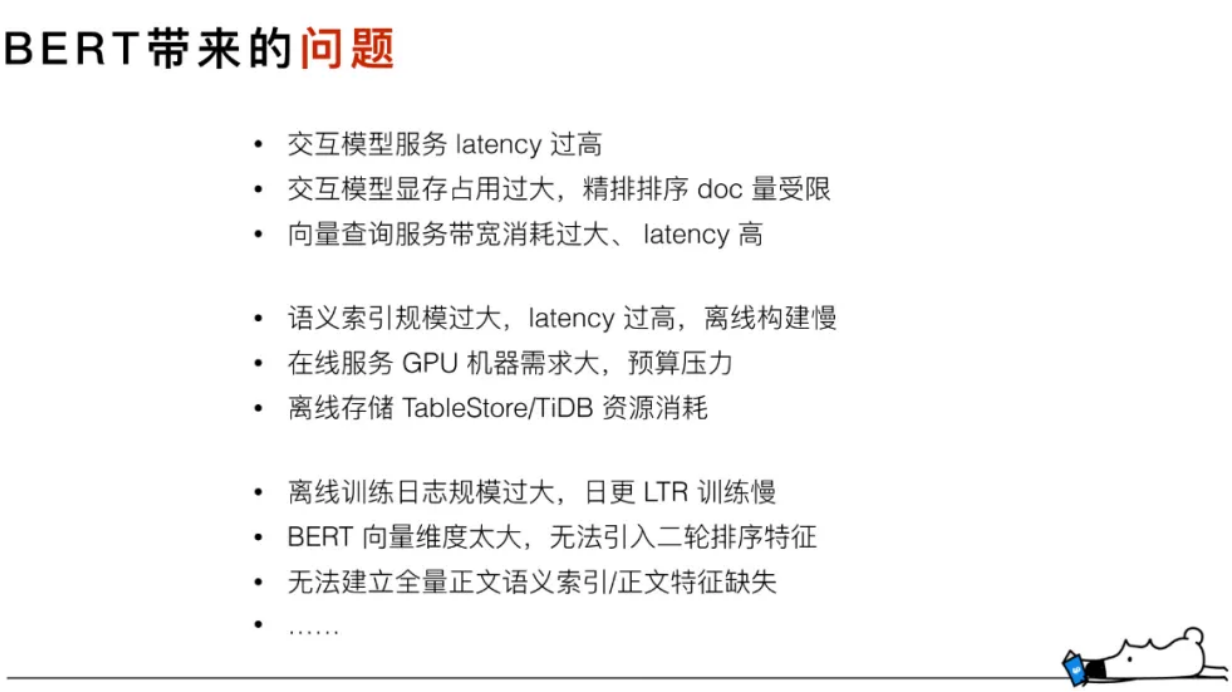
在下图中我们可以看到，BERT在知乎搜索业务的召回和排序阶段都扮演了比较重要的角色。交互模型的主要服务于二轮精排模型，依赖于线上实时的计算query和doc，为精排模块提供相关性特征。表示模型又分为在线和离线两块，在线表示模型实时的为用户输入的query提供句向量表示，离线表示模型为库中的doc进行批量句向量计算。一方面，doc向量通过TableStore/TiDB 和Redis的两级存储设计，为线上排序做查询服务；另一方面，使用 faiss 对批量doc 向量构建语义索引，在传统的term 召回基础上补充向量语义召回。

2. BERT表示模型语义召回

下面详细介绍下我们的语义召回模型。首先看个例子，对于「玛莎拉蒂 ghlib」这个case，用户真正想搜的是「玛莎拉蒂 Ghibli」这款车，但用户一般很难记住完整的名称，可能会输错。在输错的情况下，基于传统的term匹配方式（Google搜索的例子）只能召回“玛莎拉蒂”相关的 doc，而无法进行这辆车型的召回，这种场景下就需要进行语义召回。更通用的来说，语义召回可以理解为增加了字面不匹配但是语义相关的 doc 的召回。

语义召回模型整体是BERT 相关性任务中双塔表示模型的一个应用。BERT做为encoder来对query和doc进行向量的表示，基于faiss对全量 doc 向量构建语义索引，线上实时的用query向量进行召回。这个策略上线后，线上top20 doc中语义召回doc数量占总召回 doc 数量的比例能到达 5%+。

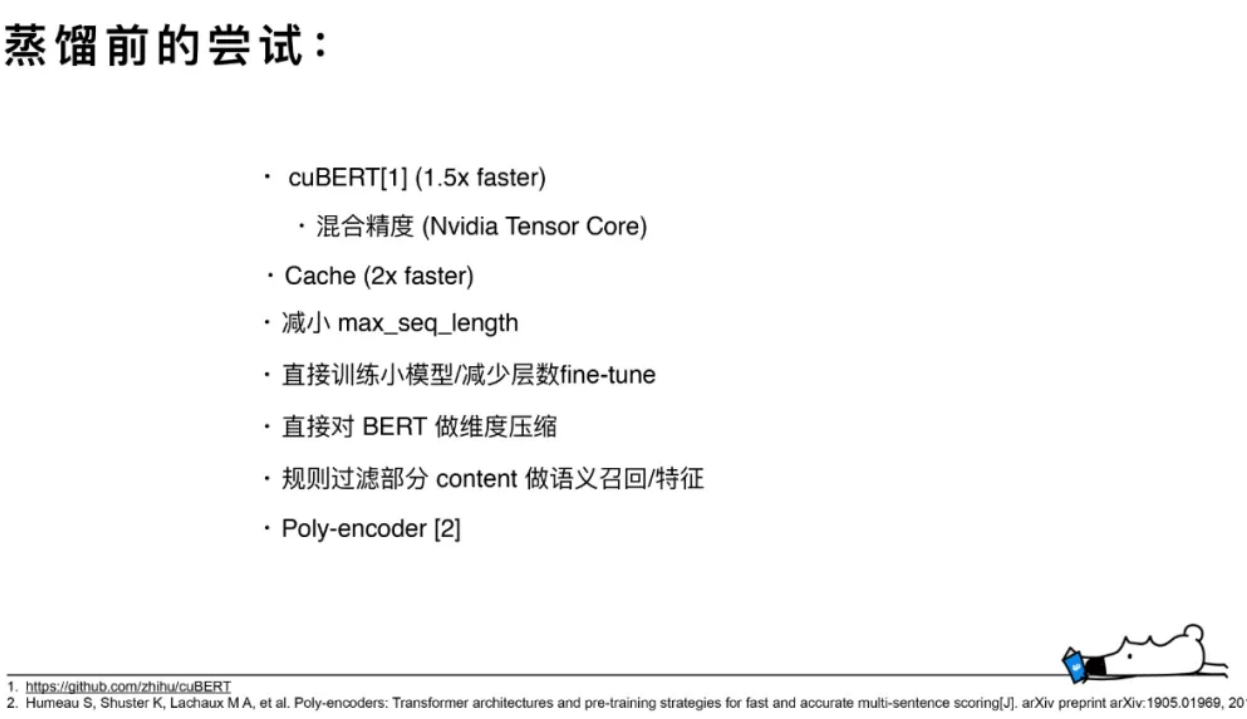
3. BERT带来的问题



BEER 模型上线后，为不同的模块都取得了不错收益的同时，也给整个系统带来了不少问题。这些问题整体可以归结为线上实时计算、离线存储、模型迭代三个方面。具体的见上图。

4. 蒸馏前的尝试

针对上述性能或存储的问题，在对BERT 蒸馏之前，我们也进行了很多不同的尝试。



1. https://github.com/zhihu/cuBERT
2. Humeau S, Shuster K, Lachaux M A, et al. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring[J]. arXiv preprint arXiv:1905.01969, 2019.

BERT 交互模型的部署放弃了使用原生TF serving，而是在cuda 的基础上用c++ 重写了模型的加载和serving，加上混合精度的使用。在我们的业务规模上，线上实时性能提高到原来的约 1.5 倍，使BERT交互模型满足了的最低的可上线要求。在这个基础上，对线上的 BERT 表示模型增加 cache，减少约 60% 的请求，有效减少了GPU 机器资源的消耗。

另一个思路是尝试给BERT在横向和纵向维度上瘦身。横向上，一方面可以减小serving 时max_seq_length长度，减少计算量；另一方面可以对表示向量进行维度压缩来降低存储开销。这两种尝试在离线和在线指标上都有不同程度的损失，因此被放弃。纵向上，主要是减少模型的深度，即减少 transformer层数。这对于显存和计算量都能得到显著的优化。前期尝试过直接训练小模型，以及使用BERT-base若干层在下游的相关性任务上进行fine-tune。这两种方案，在离线指标上的表现就没法达到要求，因此也没有上线。

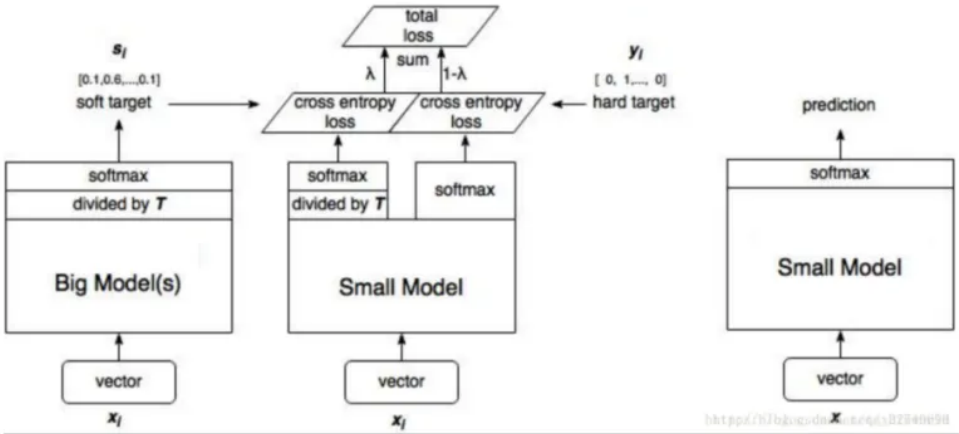
针对 doc数量过大，存储开销过大和语义索引构建慢的问题。在这方面做了一个妥协的方案：通过wilson score 等规则过滤掉大部分低质量的 doc，只对约 1/3 的doc 存储表示向量和构建语义索引。该方案会导致部分文档的相关性特征存在缺失。对于表示模型存在的低交互问题，尝试Poly-encoder（Facebook方案）将固定的 768维表示向量转为多个head的形式，用多个head做attention的计算，保证性能在部分下降的前提得到部分精度的提升。

03

智知识蒸馏及常见方案

1. 知识蒸馏

知识蒸馏



知乎

Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.

下面简单介绍下知识蒸馏。从下图中看，我们可以把知识蒸馏的整体形式简化为：大模型不考虑性能问题尽量学习更多的知识（数据），小模型通过适量的数据去高效地学习大模型的输出，达到一个知识迁移的效果。实际 serving 使用的是小模型。

知识蒸馏

- **Soft target *vs* Hard target**
  - Label Smoothing
  - Label Augmentation
- **Temperature**

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Hard target

cow	Dog	cat	Car
0	1	0	0

Soft target

cow	Dog	cat	Car
0.001	0.9	0.009	1E-06

Zi/T

cow	Dog	cat	Car
0.05	0.6	0.035	0.005

Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.

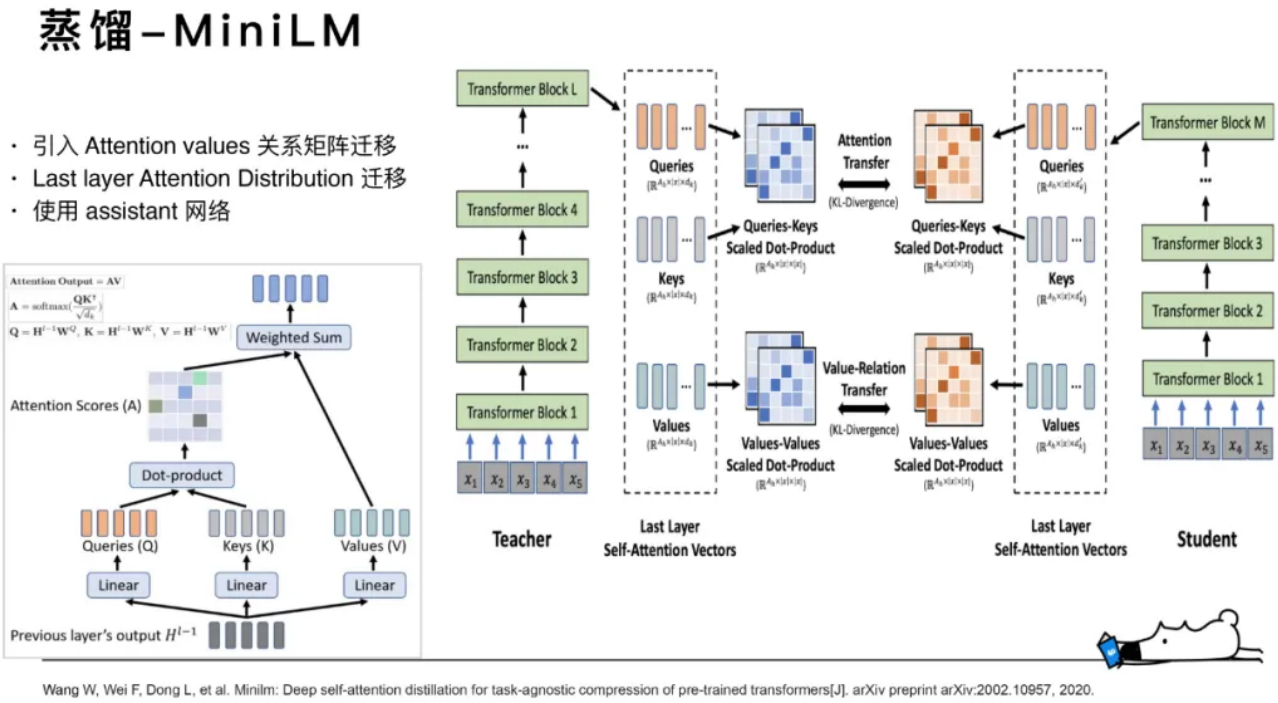
知识蒸馏为什么能有效？关键点在于 soft target 和 temperature。soft target对应的是teacher模型的输出，类似于概率分布，知识蒸馏从hard target转为soft target的学习有利于模型更好的去拟

合标签，引入temperature则是为了进一步平滑标签，让模型去学习到类别和类别中的知识。这里需要注意的是，temperature 的选取不宜过大，太大的 temperature 会导致不同类别之间的差异被完全平滑掉。

2. BERT蒸馏方案

对与BERT的蒸馏我们做了大量的调研，并对目前主流的蒸馏方案做了归纳分类。基于任务维度来说，主要对应于现在的pretrain + fine-tune 的两段式训练。在预训练阶段和下游任务阶段都有不少的方案涉及。技巧层面来分的话，主要包括不同的迁移知识和模型结构的设计两方面。后面我会选两个典型的模型简单介绍一下。

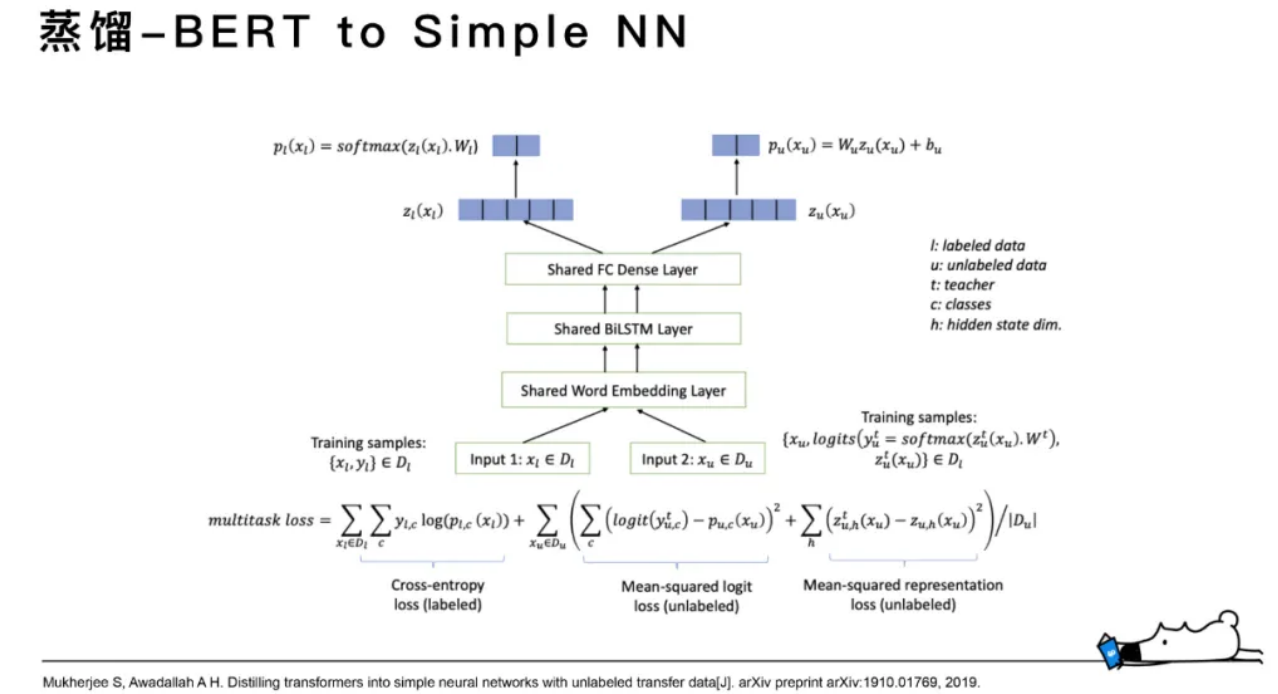
3. 蒸馏-MiniLM



Wang W, Wei F, Dong L, et al. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers[J]. arXiv preprint arXiv:2002.10957, 2020.

MiniLM是基于预训练任务的蒸馏，其是一种通用的面向Transformer-based预训练模型压缩算法。主要改进点有三个，一是蒸馏teacher模型最后一层Transformer的自注意力模块，二是在自注意模块中引入 values-values点乘矩阵的知识迁移，三是使用了 assistant 网络来辅助蒸馏。

4. 蒸馏-BERT to Simple NN



Mukherjee S, Awadallah A H. Distilling transformers into simple neural networks with unlabeled transfer data[J]. arXiv preprint arXiv:1910.01769, 2019.

BERT to Simple NN更多的是做了一些loss形式的设计，使其训练方式更高效。

04

知乎搜索再BERT蒸馏上的实践

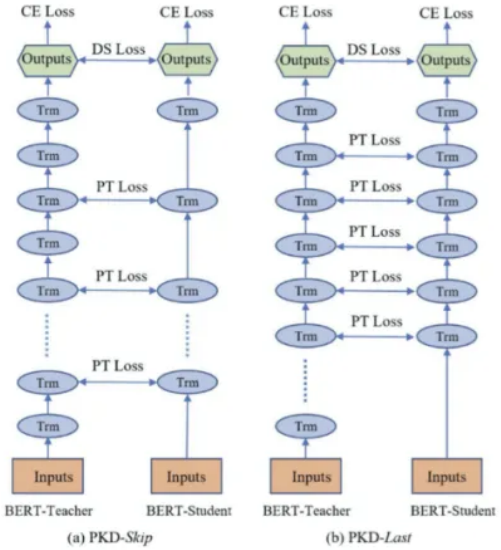
1. BERT蒸馏上的实践和收益

前面的介绍中我有提到，在做 BERT蒸馏前其实已经做了很多尝试，但是多少都会有精度的损失。因此，我们做蒸馏的第一目标是离线模型对比线上 BERT精度无损。但对BERT-base 直接进行蒸馏，无论如何都没办法避免精度的损失，所以我们尝试用更大的模型（比如BERT-large/Robert-large/XLNET）来作为 teacher 进行蒸馏。这些多层的模型均在我们知乎全量语料先做pretrain，再做fine-tune，得到微调后的模型再做蒸馏。

2. 蒸馏-Patient KD

蒸馏–PatientKD

- 策略:
  - 下游任务蒸馏
  - PKD-skip / PKD-last
  - BERT base 初始化
- Loss 设计:
  - LCE : student 的预测与真实标签的交叉熵
  - LDS : student 与 teacher 的预测的交叉熵
  - LPT : 隐藏层 normalized MSE



$$L_{PKD} = (1-\alpha)L_{CE}^{s} + \alpha L_{DS} + \beta L_{PT}$$

Sun S, Cheng Y, Gan Z, et al. Patient knowledge distillation for bert model compression[J]. arXiv preprint arXiv:1908.09355, 2019.

我们对交互模型和表示模型都做了蒸馏，主要采用了Patient KD模型的结构设计，Student模型基于BERT-base的若干层运用不同的策略进行参数的初始化，去学习Robert-large大模型的方案。

其中知识迁移主要有三部分：student的预测与真实标签的交叉熵、student与teacher的预测的交叉熵和中间隐层的向量之间的normalized MSE。

3. BERT交互模型蒸馏

BERT交互模型蒸馏

- 基于 Patient-KD 方法，直接蒸馏 24 => 6（BERT base 隔层初始化）
- 实验助教 24 => 6 => 3 better than 24 => 3
  - （助教为 BERT base last 6 层初始化）
- 训练数据：标注数据 & 随机采样无标注数据
- 迁移知识: hidden layer logits + final logits
- Point-wise loss: RMSE/Cross entropy/Cosine

Teacher	Student	nDCG@10
Robert-large	-	0.914121
-	BERT-base	0.907743
-	BERT-6L	0.903115
BERT-base	BERT-6L	0.905856
Robert-large	BERT-6L	**0.911133**
Robert-large	BERT-3L	0.904888

对于我们选的teacher模型Robert-large，单纯预训练模型其nDCG指标为0.914，线上之前使用的BERT-base 是0.907，若对BERT-base的若干6层直接去做fine-tune能达到的最高指标是0.903，对比于BERT-base精度会损失很多。

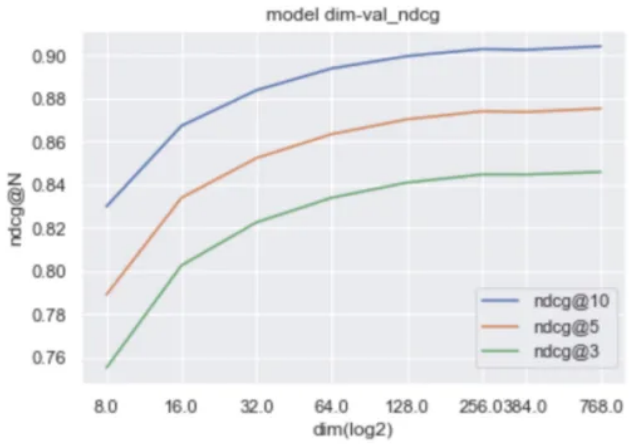
我们这块做了一些尝试，基于Robert-large从24层蒸馏到6层的话能到0.911，能超过线上BERT-base的效果。

训练数据方面，我们经历了点击日志数据挖掘到逐渐建立起完善的标注数据集。目前，相关性任务训练和蒸馏主要均基于标注数据集。标注数据分为 title和 content两部分，Query 数量达到 10w+ 的规模，标注 doc 在 300w ~ 400w 之间。

4. BERT表示模型蒸馏

BERT表示模型蒸馏

dim	val_acc	ndcg@5
base	0.8516	0.8651
64	0.8749	0.8635
128	0.8910	**0.8703**
768	0.9011	0.8753



维度压缩指标趋势图

- 蒸馏的同时维度压缩
- 交互模型作为 teacher 蒸馏
- Pairwise loss: teacher 差值拟合

$$\sum_{ij}(P(S_i - S_j) - P(T_i - T_j))$$

在BERT表示模型上，蒸馏时我们希望对向量维度和模型层数同时进行压缩，但蒸馏后得到的student模型表现不及预期。所以最后上线的方案中，表示模型层数还是维持了12层。在蒸馏时，为了提高精度，选取交互模型作为teacher进行蒸馏。因为交互模型是query和doc之间的打分，交互模型得到的logits与表示模型点乘后的打分在数量值会有较大差值，所以用pairwise形式通过teacher差值拟合来进行loss的计算。

在维度压缩方面我们做了对比实验，BERT模型输出做 average pooling 后接全连接层分别压缩至8维到768维。如图所示，128维和64维的表现跟768维差别不大，在上线时选择维度为64和128进行尝试，两者在线上表现没有太明显的差异，最终选择了64维的方案，把模型的维度压缩了12倍，存储消耗更低。

5. 蒸馏的收益

蒸馏的收益主要分为在线和离线两部分。

蒸馏的收益

Online

交互模型

- 排序相关性特征 P95 减少为 1/2，搜索入口下降 40ms
- 服务 RTX 2080Ti 8 卡 GPU 机器数减少一半

表示模型

- 语义索引存储规模 title 减少为 1/4、content 较少为 1/6
- 语义索引召回 P99 title 减少为 1/3，content 减少为 1/2
- 向量查询服务 P95 约降为 1/4, Redis 存储约较少为 1/5
- 扩充全量 content 数据语义索引和特征服务，次日留存 +0.17%
- con完全替换掉content KNRM/Pyramid，节省 GPU 资源

Offline

表示模型

- 日常语义索引构建时间减少为 1/4
- TableStore/TiDB 存储变为原来的 1/6
- LTR 训练数据减少为原来的 1/4
- LTR 日常训练时间 10h => 5h
- 粗排模型引入 32d 向量特征，提高粗排精度
- 精排引入 BERT 向量 End2End 训练，满意点击比 +0.16%

在线方面：

交互模型的层数从12层压缩到6层，排序相关性特征P95减少为原本的1/2，整体搜索入口下降40ms，模型部署所需的GPU机器数也减少了一半，降低了资源消耗。

表示模型语义索引存储规模title减为1/4，content维度从768维压缩至64维，虽然维度减少了12倍，但增加了倒排索引doc的数量，所以content最终减为1/6，

语义索引召回也有比较大的提升，title减少为1/3，content减少为1/2。精排模块需要线上实时查询离线计算好的向量，所以查询服务也有提升。

离线方面：

表示模型语义索引的构建时间减少为1/4，底层知乎自研的TableStore/TIDB存储减为原来的1/6，LTR训练数据和训练时间都有很大的提升，粗排早期用的是BM25等基础特征，后来引入了32维的BERT向量，提升了精排精度。

今天的分享就到这里，谢谢大家。