基于向量的深层语义相似文本召回？你需要bert和faiss

原创 每天都要机器学习 每天都要机器学习 1周前

在文章大话知识图谱--聊聊智能客服中，我介绍了智能客服系统利用FAQ问答库做自动问答，也就是基于信息检索的自动问答系统。它的一般做法流程是：

1. 构建一个大型的FAQ问答库，形式是（question，answer）这样的问答对；
2. 当有用户的query进来后，首先会在FAQ库中做一个粗召回，也就是召回潜在的和query意思相同的question 列表，一般使用的是倒排索引技术（elstiscsearch）；
3. 然后使用相似度算法计算question 列表中的每一个question和query的相似度，做精排序，在文章我介绍过可以使用文本相似度交互式匹配模型如MVLSTM、esim等来做；
4. 最后，如果question 列表中有和query的相似度达到预设置的阈值时，那么就把该question所对应的answer返回给用户，以回答用户的咨询。

整个流程可以简化成**粗召回**和**精排序**两个步骤，可以看到如果在粗召回阶段没有召回到真正和 query 意思相同的 question 的话，那么精排序再怎么排也不会排出能回答用户问题的 answer 了；因此我们在做粗召回的时候应该尽量多召回一些相关的question，这叫宁错召一千，不漏放一个。但是如果真的召回太多的话，那么在精排序阶段的计算量就比较大了，这在系统中体现的是计算时间超时，用户已经等得不耐烦了。因此我们需要在粗召回的数量和质量里面做一些折中。对于有选择困难症的人来说，做这样的折中总是很困难的，那么有没有方法避免折中情况呢？答案当然是有的，比如你把召回算法进行改进，让召回的 question 的质量总是比较好，那就大可不必实行“宁错召一千，不漏放一个”策略了。

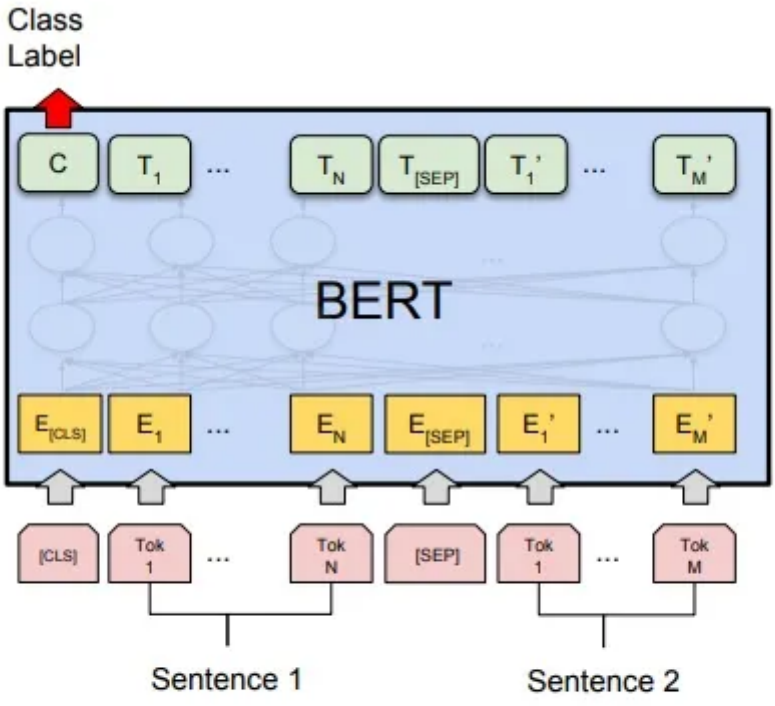
本文要讲的就是如何提升召回质量的问题。

我们知道一般的文本召回其实就是通过关键词进行召回的，通过对 query 进行分词，然后再和 FAQ 库中的 question 进行一一匹配，看和哪些question的匹配度最高，然后将其召回就好。为了提升召回速度，工作中一般借助 elstiscsearch 这个数据库来存储FAQ，然后再进行索引召回。elstiscsearch使用的其实就是倒排索引技术，它事先把所有 question 进行分词，然后建立*词-文档*矩阵，最终实现根据单词快速获取包含这个单词的文档列表之目的；倒排索引我就不做具体介绍了，有兴趣的自行去搜索资料了解。使用该方法做相似文档召回的优势很明显，实现简单、不需要训练模型、低资源需求、检索速度快，深受各大小公司喜爱。然而它的缺点也很明显，文本是具有语义的、是有语法结构的，倒排索引忽略了语句的语法结构，同时也无法解决一词多义和同义词的问题，也就它无法对 query 进行语义层面的召回。

那么如何做到从语义层面对相似文本进行召回呢？在深度学习没有流行的时候，研究人员使用主题模型来进行计算文本的语义信息，常见的有LDA、LSI、PLSA等，这些都是基于概率和统计的算法，他们通过文档中词语的共现情况来对文档、词语进行向量化表达，能够做到一定的语义层面的相似度计算。而且也有开源工具来方便进行建模学习，以 LSI 模型为例，我们可以使用gensim 来对所有（question，answer）中的question 进行主题建模，但是这面临着一个问题，即我们需要选择一个主题数量 k 作为训练参数，这个参数的选择完全看运气；另外这类模型对长尾分布的question不能很好的进行表示，在语义层面也只能做浅层的语义表达。LSI是个无监督算法，这是它的优势，我们通常将其作为文本分类或文本相似性任务中给数据打标签的辅助工具。

在文章Deep text matching--盘点11个文本匹配模型 中我介绍过可以使用表示型文本匹配模型（孪生网络、双塔网络）进行有监督训练，得到语义表示模型，然后使用该模型对所有 question 进行向量化编码，进而使用向量检索工具进行深层语义层的相似question召回。但同时文章中的实验数据也说明了，这类模型在文本匹配的效果上是比不上交互式模型的。那么有没有办法把这种孪生网络变得更强，以进行高质量的相似question 召回呢?

答案当然是使用bert嘛！所谓效果不行，使用bert。但是使用一般的 bert 是不行的，如下图所示



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

在做文本匹配任务时，通常是将两个句子进行拼接输入，然后将其作为一个二分类任务来微调。拼接方式是[CLS] sent_a [SEP] sent_b。

之所以说这样是不行的，因为模型无法单独获取 sent_a 和 sent_b 的句向量表达。原因在于多头 attention 会把 sent_b 的信息编码到 sent_a 之中，把sent_a 的信息编码到 sent_b 之中，也就是这种做法不适合用来对（question，answer）中的question进行单独编码存储。于是有研究人员自然想到使用 bert 来来搭建孪生网络[1]，如下图所示，使用两个bert分别对sent_a 和 sent_b 进行编码，然后得到句子向量之后计算余弦相似度。

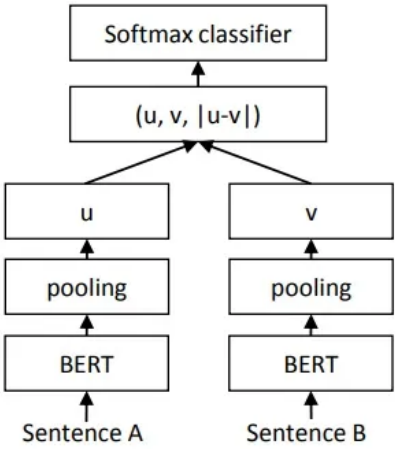


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

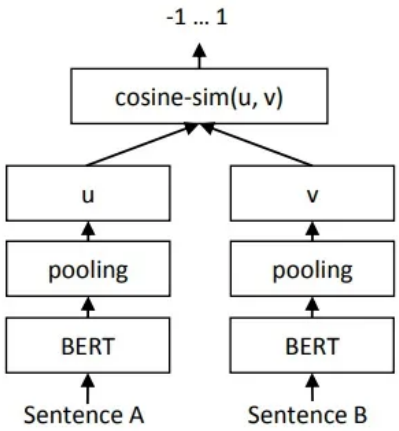


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

这样训练出来的模型就可以单独对一句话进行向量表达了，比如我要得到sent_a 的向量，那么就把 sent_b 置为空字符串就行，因为不管 sent_b 是什么都不影响模型对 sent_a 的最终表达。然而，这个模型明显太复杂了，平时训练一个 bert 机器就很吃劲了，这还训练两个 bert？而且在推理阶段我们也不能忍受多余的一个bert带来的时间消耗。那么有没有更好的模型呢！？

我们看看 UniLM 模型[2]，它是一个融合 NLU 和 NLG 能力的 Transformer 模型，是微软在2019年5月份提出来。下图是该模型的主体框架。

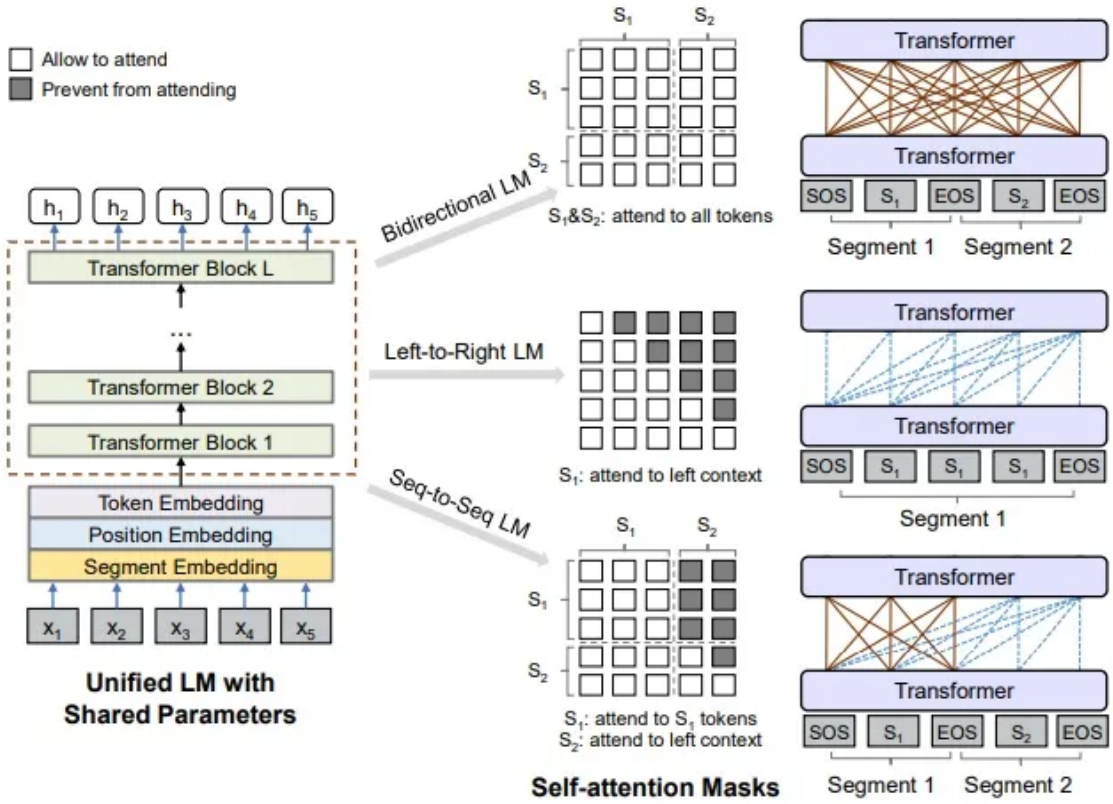
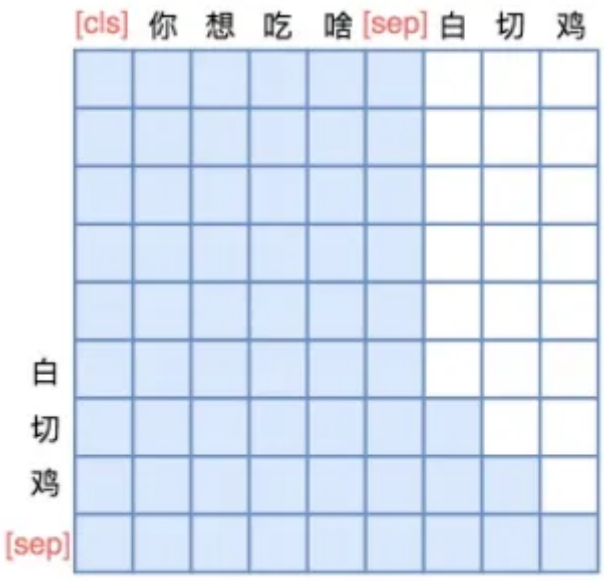
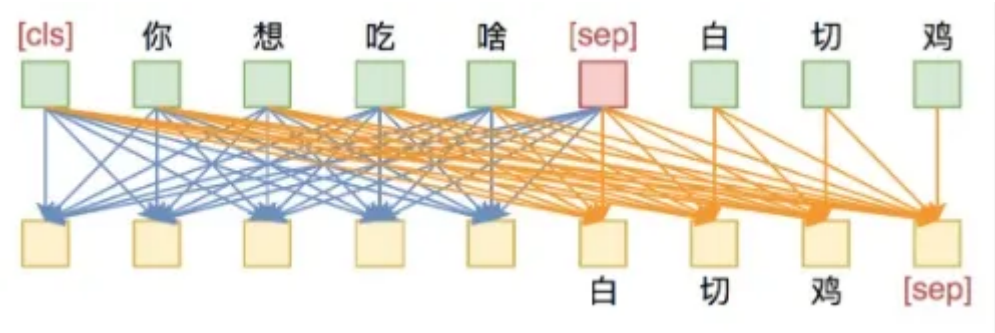


Figure 1: Overview of unified LM pre-training. The model parameters are shared across the LM objectives (i.e., bidirectional LM, unidirectional LM, and sequence-to-sequence LM). We use different self-attention masks to control the access to context for each word token. The right-to-left LM is similar to the left-to-right one, which is omitted in the figure for brevity.

下文借鉴苏建林大佬的文章中[3]相关内容。UniLM的核心是通过特殊的Attention Mask 机制来赋予模型具有 Seq2Seq 的能力。假如输入是"你想吃啥"，目标句子是"白切鸡"，那 UNILM 将这两个句子拼成一个：[CLS] 你 想 吃 啥 [SEP] 白 切 鸡 [SEP]，然后接如图的Attention Mask：

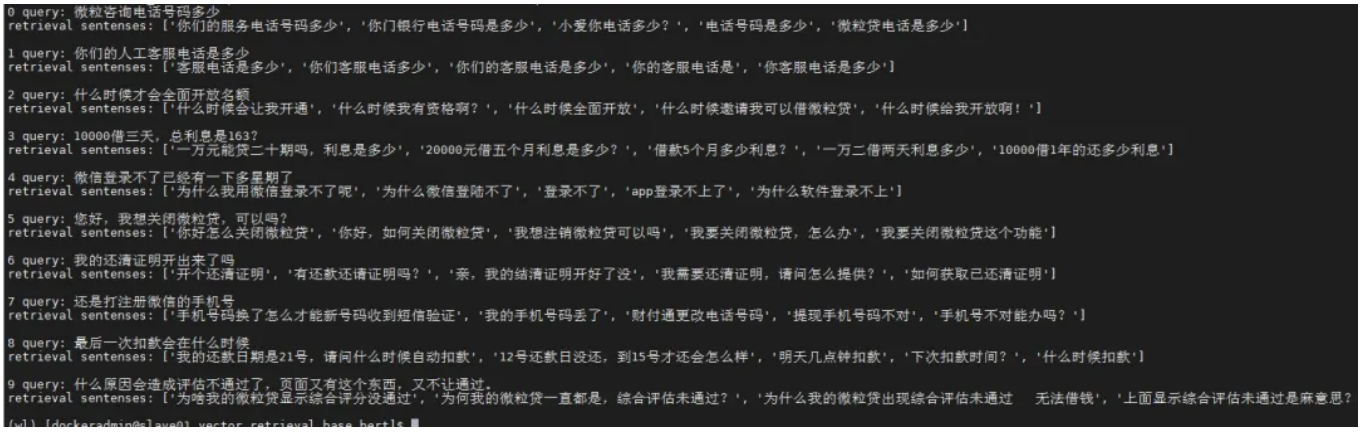


换句话说，[CLS] 你 想 吃 啥 [SEP]这几个 token 之间是双向的 Attention，而白 切 鸡 [SEP]这几个token则是单向 Attention，从而允许递归地预测白 切 鸡 [SEP]这几个token，所以它具备文本生成能力。



因为UniLM特殊的Attention Mask，所以[CLS] 你 想 吃 啥 [SEP]这6个token 只在它们之间相互做Attention，而跟白 切 鸡 [SEP]完全没关系，这就意味着，尽管后面拼接了白 切 鸡 [SEP]，但这不会影响到前6个编码向量。再说明白一点，那就是前6个编码向量等价于只有[CLS] 你 想 吃 啥 [SEP]时的编码结果，如果[CLS]的向量代表着句向量，那么它就是你 想 吃 啥的句向量，而不是加上白 切 鸡后的句向量。

我们可以看到，虽然UniLM的输入也是两个句子，但是却通过特殊的Attention Mask机制，使得模型能单独得到 sent_a 的向量表达，从而能够使得模型能对所有 question 进行事先编码成向量进行保存，从而使得使用向量进行深层语义相似性检索成为可能。我使用该模型在蚂蚁金服的数据上进行微调后，将测试集中的数据进行了向量编码，然后借助 faiss 向量检索工具进行问句的向量相似性召回，下图展示了召回的效果。



可以看到召回的相似 question 质量是相当高的。如果你对这个工作感兴趣，可以去这里看看苏大佬的开源代码[4]；当然如果你想看看我的代码也行，我会在适当的时候把我的代码提交到这个仓库[5]（可以提前点个星或者watch）。我的代码大部分都源于[4]，但是因为使用的训练数据不同，且也增加了faiss做向量检索召回，所以还是有稍微不同的。

参考资料：