谈谈文本匹配和多轮检索

原创 朱帅 AINLP 2020-03-12

收录于话题
#AINLP@我爱自然语言处理

98个

作者：朱帅
学校：华中科技大学
研究方向：文本匹配，对话系统
原文链接，点击"阅读原文"直达：
https://zhuanlan.zhihu.com/p/111769969

1. 关于文本匹配

文本匹配是NLP的基础任务之一，按照论文中的实验对传统的文本匹配任务进行分类，大致可以分为**文本检索（ad-hoc），释义识别（Paraphrase Identification），自然语言推理（NLI）以及问答匹配（QA）**。除此之外，像实体消歧等其他任务都可以采用文本匹配的思路来解决。不同的文本匹配任务，虽然实现的目标有所不同，但是从模型层面都是大同小异的，针对不同任务的模型大多是可以通用的，只不过效果上可能会有所差异。

文本匹配任务的目标是：给定一个query和一些候选的documents，从这些documents中找出与query最匹配的一个或者按照匹配度排序；本文将两个待匹配的文本定用 `text_left` 和 `text_right` ，前者表示query文本，后者表示documents中一个候选文本。

传统的文本匹配任务还是采用**基于特征的方式**，无非就是抽取两个文本tf-idf、BM25、词法等层面的特征，然后使用传统的机器学习模型（LR，SVM）等进行训练。虽然基于特征的方法可解释性较好，但是这种依赖于人工寻找特征和不断试错的方法，泛化能力就显得比较一般，而且由于特征数量的限制，导致参数量受到限制，模型的性能比较一般。

2012年以来，深度学习技术的快速发展以及GPU的出现，使得人们有机会并且有能力训练大型的深度神经网络。深度学习技术开始对计算机视觉、自然语言处理等各个领域产生了冲击，作为自然语言处理的一个分支，文本匹配当然也不例外。2013年，微软

提出 DSSM (2013)，率先将深度学习技术引入到了文本检索任务中，开启了文本匹配方向的深度学习时代。

不同于传统基于特征的匹配方式，深度学习时代的文本匹配方法可以概括为两种类型：**基于表征（representation）的匹配和基于交互（interaction）的匹配**方式。

> 所谓基于表征的匹配方式，初始阶段对两个文本各自单独处理，通过深层的神经网络进行编码，得到文本的表征，然后基于得到的文本表征，采用相似度计算的函数得到两个文本的相似度。
>
> 而基于交互的匹配方式，则认为在最后阶段才计算文本的相似度会过于依赖表征的质量，同时也会丢失基础的文本特征（比如词法、句法等），所以提出尽可能早的对文本特征进行交互，捕获更基础的特征，最后在高层基于这些基础匹配特征计算匹配分数。

2. 基于表征的文本匹配

根据不同匹配方式出现的时间顺序以及当下的流行度，首先介绍基于表征的匹配方式。上文提到的DSSM (2013) 就是最早的基于表征的匹配方法，整体流程比较简单：首先基于bag-of-words得到文本的向量表征，再基于word-hashing方法降维，接着就是多层的线性层得到最终128维的文本Embedding，最后对两个文本Embedding计算cosine相似度得到相似度分数。其实，这篇论文也基本奠定了基于表征匹配的基本范式（paradigm），即**Embedding层->Encoding层->DNN层->Prediction层**。之后的基于表征的匹配大抵都是类似的流程。

2014年，微软继续提出 CDSSM (2014)，基本流程和DSSM完全一样，无非就是将MLP替换成了CNN模型，可以提取N-gram特征，不再赘述。其实，基于表征的方式可创新的地方并不多，Embedding层是固定的，Encoding层无非再加上各种char-embedding，或者entity-embedding来引入先验知识；可以稍微有点创新的就只有DNN层，但是由于表征模型从头到尾对两个待匹配文本都是独立处理的，能做的只能是怎么得到更好的表征向量，很容易想到的就是把DNN替换为RNN型网络或者后来的Attention网络；Prediction层则是寻找不同的相似度计算函数，或者直接使用一层线性层代替。

2014年，华为也提出了一种基于表征的匹配模型 ARC I (2014)，基本范式和上述结构一致。之后，还有一些基于表征的匹配方法，包括孪生网络Siamese Network (2016)以及其变种，但是在2017年之后基本就没有基于表征的模型出现了。

3. 基于交互的文本匹配

基于表征的方式简单有效，但是其缺点也非常明显。表征用来表示文本的高层语义特征，但是文本中单词的关系、句法的特征高层的表征比较难捕获，很难判定一个表征是否能很好的表征一段文本。要想能够建模文本各个层级的匹配关系，最好能够尽早地让文本产生交互。通俗来讲就是，认识的越早，两个文本对彼此的了解就可能越多。

2014年，华为在ARC I的那篇文章中，提出了 ARC II (2014) 。首先，对输入语句得到每个单词的Embedding，然后经过一层的CNN得到两个句子N-gram级别的表征（这个使用多个不同大小的卷积核，得到多个N-gram级别表征）；接着计算基于每一个N-gram级别的表征计算交互矩阵（即一个句子中某一个位置的向量和另一个句子中其他位置的向量计算相似度，可以是点积或者cosine相似度），并在channel维度上进行拼接得到3维的张量；对上面得到的3维张量采用2D-CNN，再经过max-pooling得到最终的表征；Flatten之后经过MLP得到最终的匹配分数。

从2014年开始，中科院郭嘉丰老师团队开始在文本匹配领域发力，发表了多篇经典的论文，包括MV-LSTM (2015)，MatchPyramid (2016)，DRMM (2016)，Match-SRNN (2016)等等。前两者基本是对ARC II的补充，MV-LSTM主要是使用Bi-LSTM对Embedding进行强化编码，而MatchPyramid则提出了计算交互矩阵时多种匹配模式（**Indicator, Cosine, Dot**）。

前面说过文本匹配中不同任务的模型大多是通用的，但针对不同任务的特点也可以有一些针对性的创新。上文提到的DRMM就是针对检索领域任务的特点进行了创新。传统的文本匹配大多考虑**语义匹配（Semantic Matching）**，而检索任务中的匹配通常是**相关性匹配（Relevance Matching）**，关键词在其中起到至关重要的作用。DRMM在计算得到匹配矩阵之后，采用match histogram的方式将query中每个单词的相似度的值映射到不同的bin中，`[1，1]`这个bin表示exact match，`[-1，1）`区间均匀划分得到的每个bin都表示soft match；使用直方图中的计数作为向量中每一维的值，得到每个单词编码后的向量 $\vec{q}_i$，在经过线性层得到每个单词的匹配分数；对于query中每个单词的向量，使用Term Gating网络计算权重，对上面单词的匹配分数进行加权得到最终匹配分数。

基于DRMM，CMU熊辰炎老师的团队又接连提出了KNRM (2017)和Conv-KNRM (2018)，采用核方法对直方图的匹配方式进行改进。2017年，郭老师团队庞亮老师的一篇文本匹配综述对之前的工作进行了很好的总结。

2016年之后，随着Attention机制在各个领域的的普及，如何采用花式Attention技巧来强化单词向量表征，以及文本的交互矩阵成为之后文本匹配工作的核心。比较早期的网络有aNMM (2016)，Match-LSTM (2016)，Decomposable Attention (2016)等。Attention的使用方向主要集中在两点，一个是Embedding之后的Encoding层，通过Attention来得到强化单词的表征；一个是使用交互匹配矩阵得到两

个文本align之后的Cross Attention。采用什么样的Attention函数，以及如何组合不同Attention之后的结果就成为导致各个模型性能不同的关键点。

2017年的论文BiMPM (2017) 可以说是对之前Attention的方式进行了一个汇总，也比较清晰地描述了基于交互的文本匹配模型的范式。该模型的结构分为**Word Representation层，Context Representation层，Matching层，Aggregation层以及Prediction层。**

- Word Representation层：有Word Embedding和Char Embedding经过LSTM之后的Word表征拼接得到单词表征；
- Context Representation层：使用一个BiLSTM对两个文本进行Encoding，得到每个word的前向和后向表征。
- **Matching层**：
  - **Full Matching**：每个text中单词的前向Embedding和另一个text正向LSTM最后一个hidden state向量匹配得到m维向量，每个后向Embedding和另一个text反向LSTM的最后一个hidden进行匹配得到m维向量。
  - **Maxpooling Matching**：每个text中单词的前向（后向）上下文Embedding和另一个text中单词的前向（后向）Embedding匹配，对匹配结果max-pooling，这样每个单词前向（后向）分别得到m维向量；
  - **Attentive Matching**：计算一个text中单词前向（后向）Embedding和另一个句子中每个单词前向（后向）Embedding的相似度，取相似度值作为加权向量权重得到Attentive向量，然后使用Attentive向量和该text的对应单词向量进行匹配。
  - **Max-Attentive Matching**：取另一个句子每个维度最大值作为Attentive向量，按照上面的方式进行匹配。
- **Aggregation层**：通过Matching层得到每个句子中单词的强化表征，对两个句子分别使用BiLSTM编码，将两个方向最后一个Hidden states拼接，得到最终向量。
- **Prediction层**：将上面得到的向量，经过MLP输出最终的分数。

BiMPM中这种基于交互的文本匹配网络结构是对同一类网络很好的总结，即**Embedding层 -> Encoding层 -> Matching层 -> Aggregation层 -> Prediction层。**Embedding层获取单词的表征，Encoding层对单词表征进行强化编码，Matching层对带匹配的两个文本计算交互特征，Aggregation层对不同的交互策略进行融合，Prediction层基于融合的向量计算最终的概率。关于BiMPM的4种Matching策略，论文中有详细描述，这里不做过多赘述。

2017年的论文ESIM (2017)也是文本匹配方向比较重要的一篇论文，笔者也是通过这篇论文第一次了解了Cross Attention。该论文中Attention计算方法以及计算两个表征差和点积的方式，在之后的论文中基本上随处可见。具体来说：

- 假设 text_left 中的单词 w_a 经过Encoding之后的向量为 $\mathbf{a}$，那么可以通过Attention的方式，以 $\mathbf{a}$ 为query向量， text_right 中的所有单词的表征作为key向量和value向量，那么就可以计算得到 text_left 中单词 w_a 基于 text_right 中单词的表征，记为 $\tilde{\mathbf{a}}$；
- 为了能够强化每个单词的表征，再计算两种向量的差以及逐点乘积，最终可以 text_left 中单词 w_a 的强化表征记为 $w_a = [\mathbf{a}; \tilde{\mathbf{a}}; \mathbf{a} - \tilde{\mathbf{a}}; \mathbf{a} \odot \tilde{\mathbf{a}}]$；
- 这样，针对 text_left 和 text_right 中的每一个单词都可以得到对应的强化表征；
- ESIM中将拼接后的词向量使用BiLSTM进行序列变换，再将每个句子编码后的序列向量经过MaxPooling和AvgPooling，拼接后得到句子的表征；将两个句子的局向量进行拼接，经过一层线性层，得到最终预测结果。

同年，ad-hoc子方向的一篇论文DeepRank (2017)再次结合了文本检索任务的特点。由于文本检索任务中， text_right 通常是倒排索引返回的document，属于长文本，之前的研究直接将其作为输入，一方面增加了计算复杂度，另一方面注入了很多无效信息。DeepRank考虑标注人员对文档相关性数据进行标注的流程，即*相关位置检索->确定局部相关性->聚合局部相关性输出相关性标签*，提出通过获取每一个query-term对应document中query-centric context，来避免过多无效信息和复杂计算。

2018年之前，不同的Attention函数，像Bi-linear Attention、Dot-Attention等在不同的论文中都各有运用，具体效果视不同任务有所差异。而随着Transformer (2017)的出现，self Attention和multi-head Attention成了香饽饽，也成了后面很多模型默认采用的Attention策略。2018发表的MwAN (2018)，SAN (2018)，AF-DMN (2018)等大多是对ESIM模型的补充和扩展。比较有意思的是HCRN (2018)这篇论文，提出将复数域的Hermite矩阵的思想应用到相似度矩阵的计算中，即论文中所说的Hermitian Co-Attention。

2018年年底，BERT (2018)横空出世，预训练模型开始主导NLP战场，各个方向的SOTA几乎都被BERT刷了个遍，文本匹配自然也不例外。在这样的背景下，HCAN (2019)提出在Matching层分别计算Relevance Matching和Semantic Matching，这样既可以很好地解决信息检索中exact match和soft match的问题，在其他匹配任务中也能取得很好的效果。论文采用了花式Attention和Pooling策略，但是即便如此，最终对比实验显示结果仍然略逊于BERT。

2019年，刘知远老师团队的学生发了一篇小论文，探索BERT在信息检索中的应用Understanding the Behaviors of BERT in Ranking (2019)，文中进行了几组实验，对比基于BERT几种不同融合策略的效果，结果显示与其对BERT输出的向量进行Interaction和Aggregation，还不如直接使用 [CLS] 输出向量分类效果好。笔者也做过类似的实验，不管是对输出张量采用Pooling、Concat或者Attention，在不同的数

据集上都会有一定的上下波动，即便向上波动也十分有限，直接使用 `[CLS]` 反而更加稳定方便。

综上所述，BERT以及其衍生出来的预训练模型基本上统治了当下文本匹配数据集的SOTA榜单，相信未来会有更好的文本匹配模型对BERT形成冲击，但是可以预见的是BERT的统治还将持续相当一段时间，毕竟更容易上手。

4. 关于多轮QA检索

多轮QA检索其实也属于文本匹配类型的任务，这个任务主要应用在对话场景中，在返回Answer时考虑多轮上文Query（同Utterance）。在人们的日常交谈中，通常会出现省略、代词等导致语义不明确的单轮Utterance，如果只考虑这单个文本，显然无法准确理解语义信息。所以，在进行对话检索时，为了能够更清楚地理解某个Utterance的含义，需要考虑之前轮次的Utterance，这也就是我们所要说的多轮检索（Multi-turn Retrieval）。相比于传统的文本匹配任务，多轮QA检索由于其任务的独特性，在BERT中并没有定义这样的输入，所以目前还不断有新的SOTA涌现。

笔者最先接触的多轮检索模型是2016年的这篇Multi-view (2016)。文章的思想比较简单，主要分为**Word-Sequence Model**和**Utterance Sequence Model**。

- Word-Sequence Model将相邻Utterance使用 `__sos__` 作为拼接符拼接，再使用GRU编码语义得到序列表征，同样的方式得到response的序列表征，基于两个表征计算匹配分数p_w；
- Utterance Sequence Model首先使用CNN得到所有Utterance和Response的表征，然后将Utterance表征作为序列输入GRU得到context embedding，计算这个embedding和response表征的匹配分数p_u；
- 最后将两个匹配分数组合，得到损失函数并进行训练。

2017年出现了多轮QA领域比较有代表性的论文SMN (2017)。

- 在**Utterance-Response Matching**层，计算每一个utterance和response的相似度矩阵M_1；
- 再对utterance和response使用GRU编码，得到上下文表征，再计算相似度矩阵M_2；
- 这样对于每个Utterance得到两个矩阵，将两个矩阵作为两个channel输入到CNN中；
- 最终得到每一个U-R pairs对应的交互表征v_i，对于n个Utterance则有$[v_1, \cdots, v_n]$；

- 将序列输入GRU得到编码上下文的表征，通过最后一个隐层（或者Pooling或者Attention）的方式，得到表征向量，经过线性层得到最终的匹配分数。

2018年，出现了多篇对SMN改进的论文。DUA (2018)

- 首先通过Embedding层和GRU得到每一个Utterance和Reponse编码了上下文的向量表征，维度为 `[B，L，H]`；
- 接着，将最后一个Utterance的表征拼接到所有Utterances和Response的后面，得到每一个Sentence表征的维度为 `[B，L，2H]`；
- 基于每一个Utterance和Response使用自定义的self-Attention方法，得到每一个Utterance和Response的输出维度为 `[B，L，C]`；
- 基于原始编码矩阵计算每一个Utterance和Response的匹配矩阵 M_1，基于强化之后的词向量矩阵计算匹配矩阵 M_2；
- 使用多个filters的CNN分别对两个矩阵编码得到两个向量拼接，这样对于 n 个不同的Utterances得到 $M = [m_1, \cdots, m_n]$，使用GRU强化编码，将最终的序列向量通过Attention组合；最后经过Prediction得到匹配概率。

由于2017年Transformer的出现，无可避免有文章将Transformer模块作为编码层加入到模型中。DAM (2018)将Transformer中的一层定义为Attentive Module：

- 首先得到Utterances和Response的Embedding表征，再通过Attentive Module得到每一个Sentence的强化语义表征；
- 假设第一层Embedding输出结果为 $\mathbf{U}_i^0, R^0$，Attentive Module以此为输入构建多粒度表征（即多层，每一层表示不同粒度）；
- 基于多粒度表征对应每一层计算交互矩阵 M_{self} 和Cross Attention之后的交互矩阵 M_{cross}；
- 聚合层将每一个Utterance和Response在每一层的匹配矩阵合并成3维，将多层在channel维度合并，得到2(L+1)通道，再算上Utterance维度，得到4维张量；
- 使用3D-CNN提取特征，最后使用MLP计算匹配分数。

2019年，MRFN (2019)通过将不同级别（word，char，attention）的Embedding融合来丰富Representation层。IMN (2019)在得到Utterance和Response中单词的Embedding之后，采用类似ElMo的思路，采用多层BiLSTM并使用Attention将各层表征结合；接着，将所有的Utterances拼接得到Context，计算Context和Response的Cross Attention表征；然后，将上面的Context再分隔开，使用BiLSTM对每一个Utterance编码，通过Max-Pooling和最后一个Hidden-State得到对应的表征；再将所有Uttrance表征作为序列输入BiLSTM得到Context表征，将context和response对应的Embedding拼接，使用MLP分类。

DIM (2019)提出通过融合个性化语句，来进行个性化Response的检索。Co-teaching (2019)提出一种新的联合训练策略，提升模型的泛化能力。

MSN (2019)是目前多轮QA的SOTA模型，考虑到在多轮对话场景中，之前每一个Turn相对于当前Turn的有效信息量是不同的，所以通过Selector对之前Turn的信息进行筛选。

- 首先是Word-Selector，以最后一个Utterance作为Key，与所有Utterance计算匹配矩阵，分别使用行列Max-Pooling提取最显著特征，拼接之后转化为每一个Utterance对应的相关性得分；
- 然后是Utterance-Selector，将编码后的单词向量使用Mean-Pooling得到句向量，使用余弦相似度计算最后一个Utterance和其他Utterance的匹配度，将上面两种选择器基于权重向量；
- 同时，可以考虑最后一个Utterance和邻近Utterance的平均池化得到新的Key，这样得到k个不同的Selector；
- 将不同的选择器分数拼接成向量融合，使用sigmoid函数选择是否激活每一个Utterance；
- 最后，就是传统的Self-Matching和Cross-Matching，多个匹配矩阵拼接，使用CNN提取特征，使用GRU建模上下文Utterances的关系，使用MLP分类。

多轮检索目前和预训练模型结合的探索还比较少，其实很多多轮QA的开始都是为了获取更好的向量表征，那么可以直接选择预训练模型输出的表征，融合了更丰富语义的单词表征会一定程度上提升多轮检索的表现。

5. 融合知识的检索模型

上文提到，BERT当前在文本匹配领域已经占据了举足轻重的地位，而且其统治也将持续相当一段时间。但是，BERT还不是万能的，高层的语义表征虽然能比较好的表示单词的上下文语义，但是在常识类问题上表现却比较一般。所以，为了将人类的先验知识更好地融合到检索模型中，知识图谱和信息检索的融合是一个比较不错的方向。

前几天，PaperWeekly已经有一篇文章介绍了熊辰炎老师在知识图谱和信息检索融合方面的工作，笔者也阅读了相关的论文，并写了一篇博客对这些模型的流程进行了简单概括。但是，这些模型都假设已经完成了实体对齐的过程，而针对具体图谱的实现过程中，实体对齐（包括实体链指和实体消歧）也是当前比较热门且重要的任务。

6. 指标和工具

文本匹配可以按照分类或者排序的方式训练，分类使用的损失函数是CrossEntropy，排序使用的函数则是HingeLoss。常用的评价指标有MRR和NDCG，或者是准确率和召回率等，具体含义可以参考之前的博客。

前面提到的郭嘉丰老师团队开源的文本匹配工具MatchZoo包含了一些文本匹配领域一些常见的模型，可以用来做对比实验。笔者也写过两篇博客简单介绍了MatchZoo以及其使用方式，即关于MatchZoo和MatchZoo使用的常规流程。

MatchZoo可以快速搭建文本匹配的Pipeline，但是缺点是需要封装成内部定义的数据结构，限制了整个框架的扩展性；而且，MatchZoo只定义了英文的预处理器。因此，笔者对MatchZoo进行了一些魔改，修改了底层数据封装的格式，除了能够进行文本匹配之外，还可以快速搭建文本分类、实体识别以及多轮QA检索等常见NLP任务的Pipeline。此外，还定义了中文的预处理器，并展示了一些使用的实例，开源在笔者的Github上，**需要使用的同学可以自取。**

由于本人能力有限，可能会有一些小Bug，请自行修改或者发邮件交流。

推荐阅读

AINLP年度阅读收藏清单

中文NER任务实验小结报告——深入模型实现细节

BottleSum——文本摘要论文系列解读

抛开模型，探究文本自动摘要的本质——ACL2019 论文佳作研读系列

鼠年春节，用 GPT-2 自动写对联和对对联

用 GPT-2 自动写诗，从五言绝句开始

征稿启示 | 稿费+GPU算力+星球嘉宾一个都不少