NLP.TM[15] | 短文本相似度-CNN_SIM

原创 机智的叉烧 CS的陋室 2019-07-20

Blue Skies

Lenka - Blue Skies

【NLP.TM 】

本人有关自然语言处理和文本挖掘方面的学习和笔记，欢迎大家关注。

往期回顾：

- NLP.TM | 命名实体识别基线 BiLSTM+CRF（上）
- NLP.TM | tensorflow做基础的文本分类
- NLP.TM | 再看word2vector
- NLP.TM | GloVe模型及其Python实现
- NLP.TM | 我的NLP学习之路

今天和大家分享一篇有关文本相似度的经典文章。

Severyn A , Moschitti A . Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks[C]. the 38th International ACM SIGIR Conference. ACM, 2015.

有关实现，幸运地，我在github上找到了一个方案，大家可以参考，此处我就不谈代码而主要谈论文内容啦：

https://github.com/zhangzibin/PairCNN-Ranking

懒人目录：

- 文本相似度

- 论文详解
  - 研究背景
  - 学习排序
  - 主体模型
  - 实验结果与结论

文本相似度

先来简单介绍一下文本相似度，文本相似度是NLP下的一个分支问题，用于衡量两段文本的相似度，在搜索、问答、阅读理解等方面有很广泛的应用。传统的简单方法是通过词袋模型求距离来计算，但是这种方式是针对词汇级别的，同义词等都很难识别，而后又有了embedding模型文本句向量求相似度的方法，目前在一些场景其实也有使用，但是仍旧不是最好的方法，主要因为这个相似度的大小无法主观控制，类似的内容"我想吃肯德基"和"我想吃KFC”之类的可能无法直接识别，所以尝试使用监督学习就成了一个重要思路，通过监督学习的方式就能够有效控制相似度的计算。

从这个角度，监督学习实质上是一种人为定义，然后通过构造函数逼近的方式进行计算和转化，此处，我们对一个匹配对，假设为"query-document"，我们可以认为给他们标注一个相似度，例如"0"表示不相似，"1"表示相似，然后就可以把两个文本放入模型中即可进行模型计算，这就是基于监督学习的相似度计算，其实这个思想能用在很多领域，通过构造城监督学习的方式来提升对某个问题的掌控能力，这也是监督学习目前比较流行的一个原因吧。

论文详解

研究背景

文章本身是从LTR(learning to rank)的角度去讨论的，谈及文本相似度的计算，主要讨论了基于句法和语义特征的文本相似度的优缺点，优点在于准确性不错，但是缺点在于对外部知识甚至是知识库的依赖导致运算速度等受到限制，而深度学习的方法则更具优势，结合embedding等方式能降低对外部知识，尤其是结构化知识的依赖。

学习排序

文章对LTR进行了简单的概述。

LTR(Learning to rank)是一个研究排序的具体问题，在现实中已经有广泛应用，例如推荐系统中的排序、搜索系统中的排序等，该问题的解决方法被分为3类，pointwise、pairwise和listwise。

pointwise是对每个待排序的条目进行打分，根据大分大小进行排序，现行推荐系统的CTR预估就是目前一个比较典型的pointwise方法。

pairwise是指，两两对比待排序条目，然后根据对比结果进行重排序。

listwise是指，以待排序列表整体为单位进行的排序方法。

主体模型

整个模型其实可以用文中的一张图简要表示。

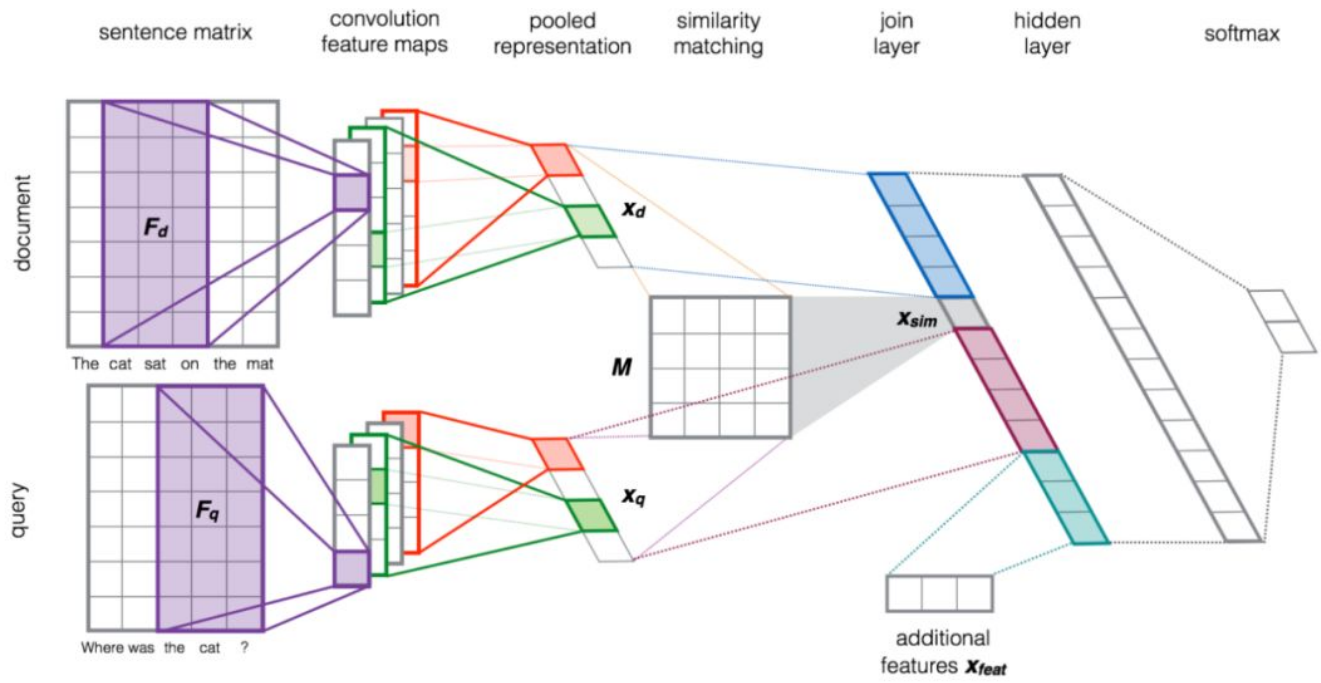

Figure 2: Our deep learning architecture for reranking short text pairs.

对于两套文本的输入，很基本的采用embedding方案将文本转化为句子矩阵(此处文章用的是w2v)。

然后用卷积+池化的方法进行特征提取，卷积一块没使用1维卷积，而是用2维卷积的方式体现bi-gram甚至tri-gram。

卷积+池化后实质上两个句子已经转为了两个句向量，句向量就可以开始进行相似度衡量了，这块应该是我觉得的从本文学到最大的点了。文中构造一个相似矩阵M，用于计算两者的相似度，这个相似矩阵M。

$$sim(\mathbf{x}_q, \mathbf{x}_d) = \mathbf{x}_q^T \mathbf{M} \mathbf{x}_d$$

然后，将计算得到的相似度、两个句向量、以及额外特征进行拼接组合，得到一个向量，这个向量内涵盖了相似度、query句向量、document句向量以及额外特征4各方面信息，通过全连接层计算后最终到达输出层，整个深度学习模型完成。

训练使用的损失函数是分类常用的交叉熵损失函数，配以L2正则，用adadelta进行训练。

$$\begin{aligned} \mathcal{C} &= -\log \prod_{i=1}^{N} p(y_i|\mathbf{q}_i, \mathbf{d}_i) + \lambda \|\theta\|_2^2 \\ &= -\sum_{i=1}^{N} [y_i \log \mathbf{a}_i + (1-y_i) \log(1-\mathbf{a}_i)] + \lambda \|\theta\|_2^d, \end{aligned} \tag{3}$$

实验结果与结论

其实实验结果本身不重要，重要的是实验结果中体现的现象以及作者的解释，根据实验结果以及作者的讨论，本模型的特点主要如下：

- 不需要手动特征工程，也几乎不需要预处理和外部资源
- 在多个准确性指标下性能提升，P@30、MAP
- 对较好的embedding方案有一定的依赖性