# Web项目报告——新闻推荐

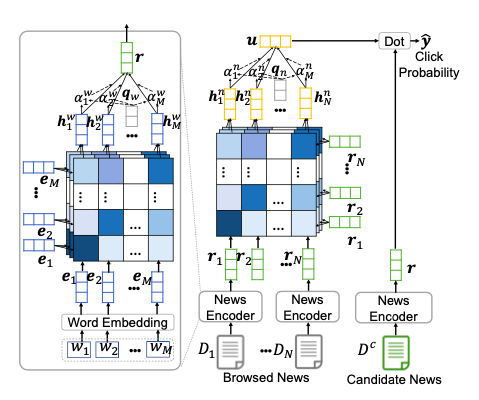
**卢淑祺 2019101377**

## 1. 项目目标：

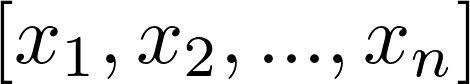
随着大数据时代的来临，信息筛选的能力越来越重要。而在媒体信息泛滥的今天，新闻推荐算法备受关注。新闻推荐是针对用户的兴趣，预测其可能点击的新闻并推荐给用户。新闻推荐可以帮助用户找到感兴趣的新闻，减轻信息过载。在新闻推荐的算法中，对新闻和用户进行精确建模对于新闻推荐是至关重要的。对用户的建模是通过对用户的历史浏览记录进行语义分析，构建用户的兴趣偏好画像。对新闻的建模则是利用新闻内容对新闻文档进行语义分析，从而可以将新闻文档的语义与用户历史的偏好进行语义匹配，在当前的候选新闻列表中选出符合用户兴趣的新闻进行推荐。整个过程涉及到用户兴趣画像的建模和新闻文档表示的生成，一方面需要考虑语义理解的准确性，一方面需要考虑线上运行的效率。

## 2. 算法介绍

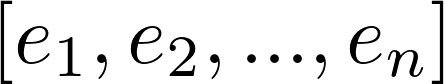
实验框架如下图所示：



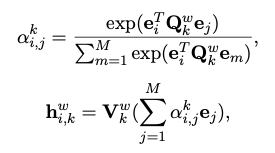
输入的文档形式为词id序列:



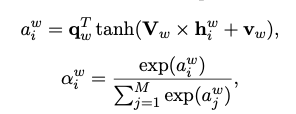
首先将文档词通过word embedding 层转换为词向量：，本次实验采用了两种不同的编码方式，一种是利用glove预训练词向量，一种是利用预训练好的bert对文档进行编码，取bert最后一层的输出作为每个位置的词向量。

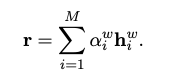


得到词向量之后计算文档的语义向量。首先经过一个self-attention层为每个词计算结合上下文的语义向量：



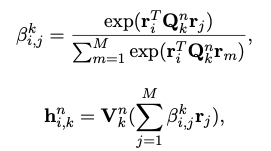
然后通过一个attention层将所有词的语义向量融合得到最终的文档向量：



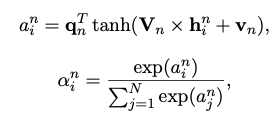


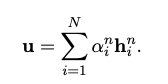
对候选新闻文档列表中的每个文档按照上述方式获得候选文档的语义向量。

对用户历史浏览记录中的每个文档都按照上述过程得到历史文档的语义向量，接下来计算用户画像。将所有历史文档的语义向量用一个self-attention层处理，得到每个历史文档结合上下文的语义向量：

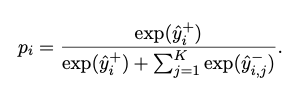


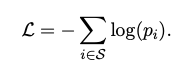
接下来通过一个attention层将所有历史文档的语义向量融合得到最终的用户兴趣画像：





最终通过向量的点积计算候选文档与用户历史兴趣的语义匹配程度,并采用交叉熵的形式进行训练：

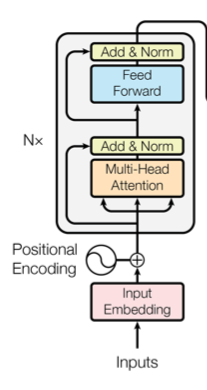




### 2.1 词编码：

Glove: 将所有新闻文档的标题作为语料，按照(https://nlp.stanford.edu/projects/glove/ )中的方法训练词向量，将word embedding 层的词向量矩阵按照训练好的词向量初始化，训练时进行fine-tune。

Bert：bert是利用英文的维基百科的语料预训练的语言模型。它的模型结构是多层transformer堆叠。单层transformer的结构如下图所示。预训练时输入是维基百科中的语句，将其中一些词替换为[mask] 掩码。训练目标则是预测掩码位置原始的词。预训练好的模型包含丰富的语义知识，能够较好地表达文本语义。这里本次实验采用与bert一致的分词器以字节为单位编码语句，将词id输入到预训练的bert层，取最后一层每个位置的输出作为每个位置的词编码。



### 2.2 关于模型的分析：

本次实验的模型对于文本处理最主要的结构是self-attention.在文本处理中常见的结构还有RNN、CNN等结构。相比于序列模型LSTM或GRU，self-attention也同样能够捕捉上下文的语义关系，但是同时每个位置的输出还可以并行计算，因此效率更高。而与CNN相比较，CNN虽然速度快，但是受限于卷积核大小，难以捕捉长距离的上下文关系，因此在处理上下文语义上，尤其是对于本次实验长时间的用户历史浏览序列，Self-attention的结构更有优势。

## 实验：

### 3.1 数据集：

数据来源是微软提供的新闻推荐数据集( https://msnews.github.io/),该网站仅提供训练集与验证集。训练集和测试集中有behaviour.tsv与news.tsv两个文件。

其中behaviour.ts中如下有四个字段：User ID: 用户编号； Time： 用户点击时间； History：用户历史点击的新闻编号；Impressions： 当前推荐给用户的新闻编号候选列表。

News.tsv中有如下四个字段：News ID: 新闻编号； Category： 新闻种类； Title： 新闻标题； Abstract： 新闻摘要.

由于个性化新闻推荐的数据涉及到用户隐私，该公开的数据集对用户名进行了匿名处理。 其中共有50，000个用户，用户数据为从2019.10.12到11.22约一个月的点击记录。由于其中用户历史浏览数据长度为0-300不等，分布稀疏，本次实验取平均值50，即截取50个的用户最新浏览的历史文档。同时由于每个候选的新闻列表长度最大不超过300，因此问题可作为排序问题，无需为了提升效率预过滤一些文档。同时本次实验采取文档标题作为新闻文档的内容，标题最大长度设置为10.

### 3.2 实验：

输入: CandidateNews\_id: 候选新闻文档的词编码； ClickedNews：历史点击新闻的词编码； History\_len: 每个历史新闻文档的长度（用于注意力层去掉padding的空字符）; Candidate\_len: 每个候选文档的长度： All\_history: 整个历史序列的文档个数； All\_candidate: 整个候选文档的文档个数.

输出：测试时输出的是候选文档与用户历史的匹配得分.

glove词向量设置为100维，共28929个单词。bert层使用的预训练模型是12层, 前馈神经网络的输出为128维的bert模型；训练模型时只finetune后6层。前六层的参数固定。模型中的attention层的注意力向量设置为200维。

### 3.3 实验结果:

评价指标采用的是分类和排序问题中常用的auc, mean\_mrr， ndcg@5, ndcg@10。

使用glove词向量的模型以及以一样的训练轮次，学习率等训练设置训练的bert 编码的模型在验证集上的结果为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| metric | auc | mean\_mrr | ndcg@5 | ndcg@10 |
| Glove embedding | 0.633 | 0.2915 | 0.3188 | 0.3808 |
| Bert embedding | 0.6378 | 0.2947 | 0.3208 | 0.3829 |

### 3.4 结果分析：

使用bert层编码的模型准确率只有微弱的提升. 时间上，使用glove的模型编码层只需要按照词id读取，而bert编码则有O(n\*n)的时间复杂度，n为文档的长度。空间上，使用glove编码只需保存word\_size\*emb\_dim大小的词向量矩阵，而bert编码除词矩阵以外还需要O(n\*feed\_forwar\_dim)大小的空间存储每层的中间结果。

## 4. 结论：

self-attention：可以高效率的计算长距离上下文的关系，自动提取重要的语义部分，相比于RNN和CNN有一定的优势。

数据量较小或者文本序列不长的情况下利用pre-trained bert模型可能弊大于利。