基于循环神经网络的新闻推荐

姓名：俞沛， 学号：SA19011173
姓名：蒋金刚，学号：SA19229037

一、实验内容

1．背景

当今，互联网技术的飞速发展使得纸质报纸已经被电子新闻所替代，我们每天打开手机app，就能够浏览发生在世界各地的新闻。这样获取信息快捷的方式吸引许多用户在线浏览新闻。然而每天都会产生大量的新闻文章，用户不可能通过阅读所有的内容来找到自己感兴趣的内容。因此，个性化新闻推荐对于在线新闻平台以用户兴趣为目标，消除信息过载非常重要。

2．实验定义

给定用户浏览历史数据(news1, news2, ..., newsn)，和候选新闻列表(candidate1, candidate2, ..., candidatem)，往往表现为一个页面中的内容。通过用户历史点击序列，学习出序列模式（用户表示），再对每一个候选新闻求推荐得分，用以推荐。

二、实验环境

Python 3.7，Pytorch 1.3，Cuda 10.2

三、系统实现过程

1. 关于数据

1）behaviors 是用户浏览历史数据

训练集中有 156965 条数据，一共有 33915 条新闻，50000 名用户；

测试集中有 73152 条数据，一共有 37681 条新闻 ，50000 名用户；

其中用户并不完全重合。

2）news 是包含当前训练集（测试集）中出现所有新闻的详细情况，包括标题，摘要，类别等

3）每条数据的平均点击长度为 33.225386561892186，最长为 558，最短为 1，新闻标题的平均长度为 12.355170559234903，最长为 66， 最短为 1。具体分布如图 1.。

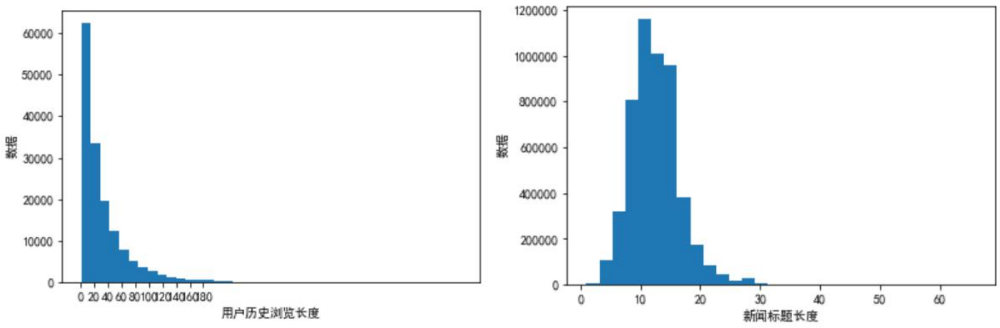


图 1

2. 数据具体处理

该问题主要针对新闻的标题所提供的语义信息，取挖掘用户的喜好，所以对于新闻标题中的单词进行处理。我们首先载入停用词，再利用 nltk 库中 word_tokenize 方法对于新闻标题进行分词，过滤到对新闻语义影响很小的停用词，再将没有见过的词纳入词典中，即将词索引化，以便后面放入 embedding 网络中；并且载入 glove 的词库，一个已经训练好的词库，

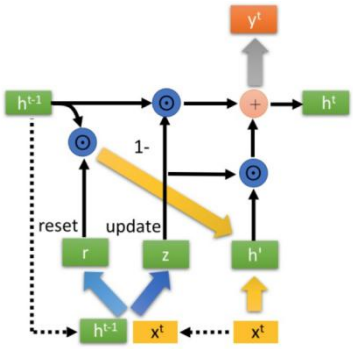
我们选择了 40 万词汇，100 维嵌入大小的词库作为后面网络 embedding 层的初始化。

为了能够放入网络模型中更好的训练，对于训练数据进行规整，我们取用户浏览平均长度 35，标题平均长度 15 作为网络输入的固定长度，对于不足的数据补 0，超过的直接截断；训练数据，对于候选新闻集合，对其中每一个正样本都携带四个负样本；验证集同改，测试集候选新闻浏览长度无法修改，固不变其他修改。

3. 模型（Double-GRU）

处理时间序列数据，首先能想到的就是 RNN 模型，所以这里我们使用了 GRU 作为模型基础。

1） GRU（Gate Recurrent Unit）是循环神经网络（Recurrent Neural Network, RNN）的一种。和 LSTM（Long-Short Term Memory）一样，也是为了解决长期记忆和反向传播中的梯度等问题而提出来的。与 LSTM 相比，GRU 内部少了一个”门控“，参数比 LSTM 少，但是却也能够达到与 LSTM 相当的功能。考虑到硬件的计算能力和时间成本。



2） Double-GRU：

如图所示，先对于标题中单词，利用第一层的 GRU 学习的那次之间的关系，并且获得该新闻的向量表征，再对于用户浏览历史中新闻，利用第二层的 GRU 去学习用户的表示。对于候选新闻，通过第一层 GRU 后学习到候选的嵌入表示，再与用户表征作点积，得出候选新闻的点击概率，用以推荐。

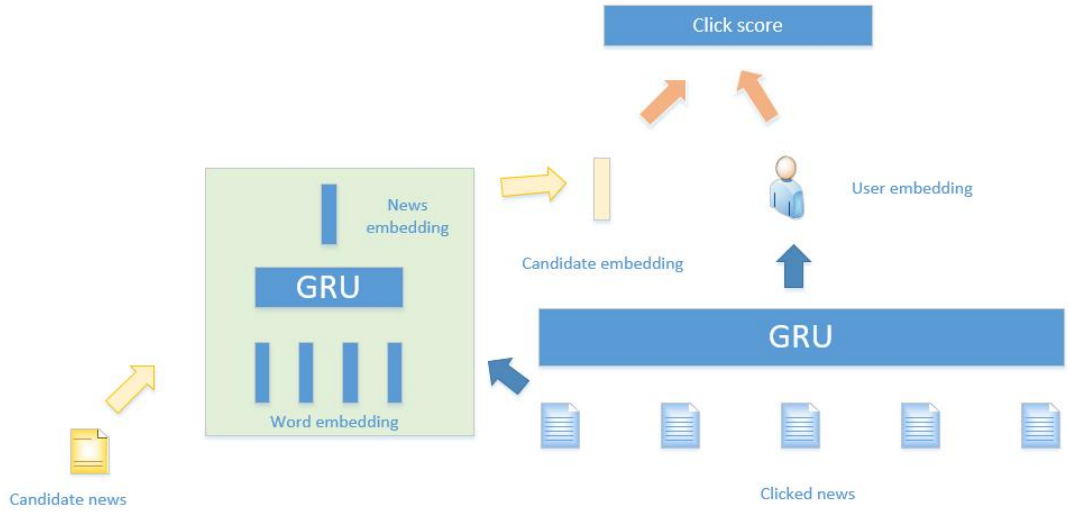


图 2.

4. 代码

model.py: 模型文件

start.py:训练和测试文件

data_process.py: 数据处理文件

详细见文件夹中的 python 文件

四、测试与结果分析

用户向量表征维度设置为 100 维，Loss 选择交叉熵，使用 Adam 优化器，学习率为 0.001。再验证集上计算准确率（是一个五分类的问题），再测试集上计算 auc（由于正负样本不平衡，单纯的准确率并不能体现模型的好坏，所以使用 auc 作为一个模型性能的参考）

1. 首先 Embedding 初始化后不更新，跑到 20 个 epoch，auc 基本稳定。loss，acc，auc 的情况如下：

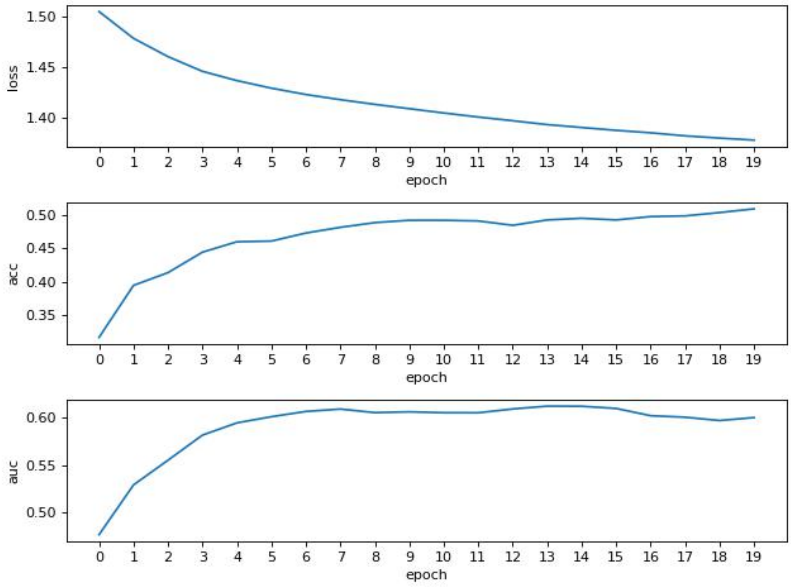


图 3.

最大的 auc 为 0.612017324259288，acc 为 0.5085。

2. 接下来改变用户表示维度，和词向量表示的维度的结果：

word_embedding_size	user_embedding_size	auc	acc
100	50	0.612017324259288	0.508
200	50	0.6081362951970157	0.485
300	50	0.6093585070289421	0.491
100	80	0.607203923507093	0.501
100	100	0.603314521323229	0.499

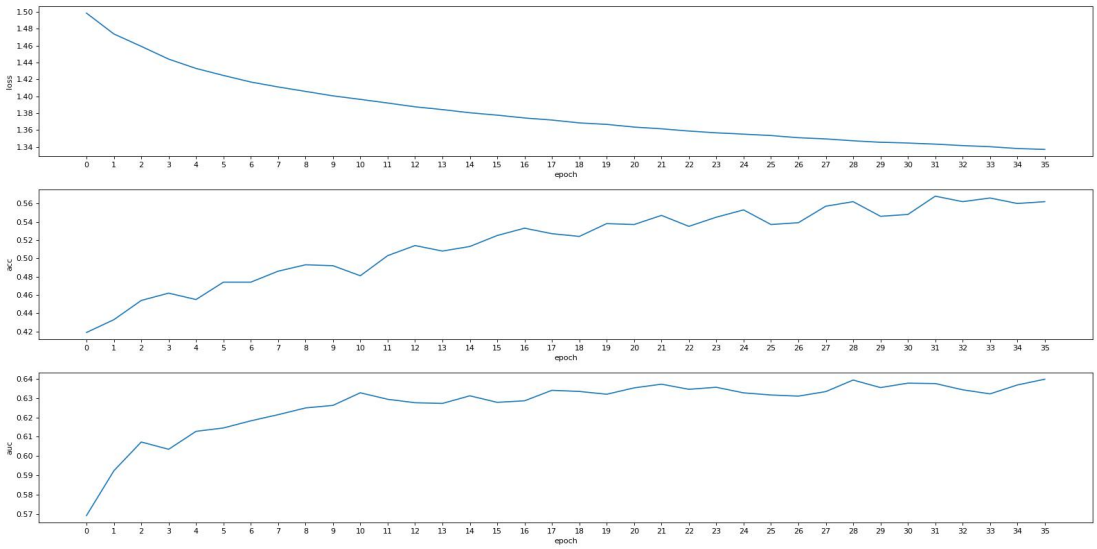
由上可以看出，用户表示维度和词向量维度对最终的结果影响不是很大。

3. 经过思考和分析，取 GRU 最后隐层输出作为新闻表示或者用户表示，似乎不妥。因为 GRU 所学习的是一种序列关系，对于词来说，这样学到的表示是关于新闻标题中词的后续，而非我们需要的整体表示，这里我有两种改进方案：

1）取各隐层输出的平均作为最后相应的表示，最终结果如下：

word_embedding_size	user_embedding_size	auc	acc
100	50	0.6398144840709209	0.568
100	100	0.6463941160002625	0.584
200	100	0.625851857325363	0.538
300	100	0.6296317336789782	0.547

可以看出效果提高 3 个百分点，效果明显。



2）对隐层输出求一个 attention 权重，通过加权和计算最终的表示：

word_embedding_size	user_embedding_size	auc	acc
100	50	0.6335648367741498	0.57
100	100	0.6375712389793835	0.584
200	50	0.6404026160450969	0.603
300	50	0.6439254926364463	0.596
300	100	0.6454557178866362	0.599

可以看出 attention 算表示的结果要比均值稍微好一点，因为 attention 考虑了词对之间确切的权重关系，通过加权求和算最终表示，能够反映出每个词表示对于新闻表示的确切的贡献，而取均值将每个词对于表示的影响都相等，显然是不符合直觉的，实验也验证了这一点。增大表示维度，对最终结果有一定的正效益。

五、心得体会

通过这次实验，比较清晰的了解了文本处理的过程，包括分词，去除停用词，引入预训练的词向量等等，还了解了 GRU 在于建模序列数据的重要作用，attention 和均值化对于序列数据学习整体表示时的增益。接下来还会尝试，对于数据处理的改进， 增加一些关于新闻类别或者实体嵌入表示的因素；对于模型的改进，尝试 CNN，还有 GRU 最热门的替代 self-attention，进一步去体会各个模型学习序列数据之间的差异。