基于历史访问数据的 PV 预测数学建模

摘　　要

在题目查询类教育应用使用量激增的今天，各大应用开发商都希望为题库的题目答案提高质量，以提高用户体验。然而题库中的题目非常多，如果能够提前一天（或几天）预测到哪些题目会成为热门题目，PV（Page View）值较高，并对这一小部分题目的答案进行高质量加工，这样就能最高效率地提升用户体验。本文主要通过建模构建了一套可行的试题 PV 预测算法，可以提前一天或几天给出对热门题目的预测。

总体上讲，我们首先利用 TF-IDF 算法结合停用词表对 OCR 识别结果和试题信息进行了关键词提取，再将所有题目根据关键词分为 14 个种类，预测出不同类别题目的 PV 值走势，再根据预测的热门类别得到相应的热门题目，实现了对热门题目的预测。最后，我们还分析了这一模型的可靠性和优缺点。

由于题目 PV 值波动主要是由学校教学进度的推进引起的，所以我们认为实际上是涉及某个知识章节的一类题的 PV 值在整体波动，而同一类别内的各个题目的相对热门程度与题目质量相关，是长期稳定的。因此单个题目 PV 值的波动其实是由该类别题目总 PV 值波动引起的。所以直接对单个题目的 PV 值进行预测并不合理。我们必须在预测之前将 1000 道题目分为不同的类别，类别的划分就相当于知识章节的划分。然后再以类为单位进行 PV 预测。两个题目之间的关联可以由共有的关键词进行表征，为此我们建立了一套依据关键词来对题目进行分类的算法。最后我们将 1000 道题目划分为了 14 类。我们认为第一步的分类和第二步的预测同样重要，甚至比预测更重要。

通过分类，我们已经建立了所得的 14 类题目在 14 天内 PV 值的时间序列。由于已有数据的时间尺度短，且波动的因素仍处于灰箱状态，综合分析考虑，我们使用神经网络模型进行预测。首先我们对数据进行标准化处理，利用前 13 天的 PV 值进行学习，通过调整迭代次数和隐含层数量，得到收敛结果，要求其与真实值吻合。然后利学习结果对第 14 天 PV 值进行预测，并将第 14 天作为验证集进行模型的检验。

最后，将第 14 天我们的预测结果与真实的结果进行比对，发现二者较符合，从而说明了我们的模型的可靠性。

KEYWORD：PV 预测、TF-IDF 算法、关键词提取、文本分类、神经网络模型

目录

1 问题的重述

1.1 研究背景

作业帮 APP 题目查询系统每天都有数以亿计的查询请求，大部分请求都是用户上传的图像。查询过程由几个次序构成：OCR 系统将图像转化为文字；NLP 系统将文字拆分为分词，将这些分词与题库中不同题目已知的分词构成比照，检索出相同试题。现在重点关注与 OCR 结果的相关性排在首位的试题。

我们把每道题每天出现在搜索结果中首位的次数记为该试题的 PV(Page View)。根据试题的 PV 排序，我们可以得知每天哪些题目是热门题目。如果我们能够提前一天（或几天）知道当天哪些试题将成为热门题目，我们就可以提前对这些试题进行高质量加工，增强用户体验。

1.2 问题提出

由于教育类应用场景具有很强的周期性，很多需求与学习进度相关，所以试题的 PV 也一定具有某种周期性。通过建立合适的数学模型，我们可以预测 PV 的变化趋势，从而提前一天（或几天）确定热门试题的走势。

2 问题分析

2.1 陈述思路

由题附数据仅能得到所给 1000 道题目 14 天的 PV 值，我们发现仅从单个题目搜索量基本看不出周期性。但是，题目的搜索需求是与学习进度有关的，因此可以认为相同类型的题目总 PV 值可以构成周期性谐波（这在之后得到了部分验证）。而单个题目的 PV 值变化有较大的偶然性，单独一道题的 PV 值波动函数在更长的时间尺度上也很难形成自相关的谐波。

我们决定先将 1000 道题目分为不同的类别，然后具体分析每一类的变化趋势。同时，我们认为包含同一知识点的题目中会含有相同的关键词，因此我们决定先从题目中提取关键词，再根据关键词进行下一步分类。

完成分类后即可对不同类别题目分别进行 PV 值预测。

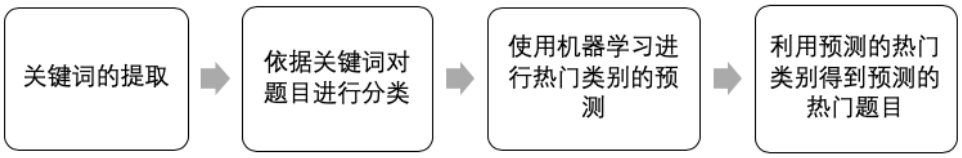

图 1: 总体思路流程图

2.2 分类过程的初步简化与去噪

首先，OCR 识别结果文字与题库中的题目文字拆分后的分词均可以与一个正整数集合构成一一映射关系，从而简化表达。这步操作在题目所附数据中已经得到体现。

其次，自然语言中存在一些超高频的词语，包括英文字符、数字、数学字符、标点符号及使用频率特高的单汉字等，这些功能词（又称停用词）很少单独表达文档相关程度的信息，在检索过程中应直接进行舍弃。这样一来可以提高检索效率，二来可以避免噪声干扰。我们一方面可以通过已有的算法（如 python 的 jieba 库）直接判断出哪些分词是停用词；另一方面也可以利用题目所附停用词表进一步筛选出停用词。

2.3 对预测手段的选择

在网络上进行一番搜索后，发现已有不少工作与访问量（如某网页访问量，某客服电话接听数）预测相关，也有很多现有的高级算法（如 Prophet, TBATS, ARIMA, snaive），但是它们进行预测时利用的数据已经可以看出存在明显的周期性 [1]。而由题附数据仅能得到不同类别题目 14 天的 PV 值。

另外，由于 14 天这个时间尺度很小，我们几乎无需考虑数据整体变化趋势（如由用户量的增多造成）或者更长时间尺度上不固定频率的上升和下降（一般由商业因素造成）[2]，在简单归一化操作后认为它仅存在谐波特点。其次，问题所要求的预测周期并不高，无需进行长期预测。综合分析考虑，我们使用神经网络模型，可以仅对未来一天（或几天）不同类题目的 PV 值进行预测，预测精度高，再进一步得到预测的热门试题。

3 模型假设与符号

- 题目 PV 值波动主要是由学校教学进度的周期性推进
- 教学进度的推进引起的是某一类别题目整体 PV 值的波动
- 同一类别的不动题目的 PV 值相对大小只与题目本身质量相关，不随时间长期波动
- 单个题目的 PV 值由所处类别的波动情况和在该类别中的相对热门程度决定
- 即有很多题目都同时包含这两个关键词时，我们可以认为包含这两个关键词中的任何一个的题目都属于同一类别。

文中涉及的符号规定如下：

符号	意义
q	一个题目
S_q	题目 q 的所有关键词 k 的集合
k	一个关键词
S_k	所有以 k 为关键词的题目 q 的集合
r_{ij}	关键词 k_i 和 k_j 的关联程度
x_m	认为两个关键词有关联的临界关联系数 r_{ij}
y_m	按照关键词整合分类时允许的类别中的最小关键词数

4 模型的建立与求解

4.1 题目关键词的提取

考虑到题目的搜索需求是与学习进度有关的，即与知识点的学习进度有关，因此我们认为题目搜索量的变化趋势与题目所含知识点的搜索量变化趋势是相同的，从而可以将 1000

个题目根据知识点进行分类，然后具体分析每一类的变化趋势。同时，我们认为包含同一知识点的题目中会含有相同的关键词，因此我们决定先从题目中提取关键词，再根据关键词进行下一步分类。

对于题目中关键词的提取，我们采用了 TF-IDF 算法与停用词表结合的方法。对于 TF-IDF 算法，我们首先计算每个词的词频 (即 TF)，有

$$\text{词频 (TF)} = \frac{\text{某个词在文本中的出现次数}}{\text{文本总词数}}$$

但是仅有词频并无法完全反应一个词的重要程度，因此我们再计算另一个调整系数——逆文档频率 (即 IDF)，有

$$\text{逆文档频率 (IDF)} = \log(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数 } +1})$$

最后我们可以计算出 TF-IDF，有

$$\text{TF-IDF} = \text{词频 (TF)} \times \text{逆文档频率 (IDF)}$$

利用得到的 TF-IDF 进行排序即可实现对关键词进行提取。在具体实现中，我们首先将每个题目在 14 天中所有的 OCR 识别结果进行整合，得到了 1000 个文本 (代码见附件中的 sort.py)，之后采用了 python 中的 jieba 库来实现 TF-IDF 算法，将 14 天的 OCR 识别结果作为语料库，结合所给的停用词表，对每个题目的所有 OCR 识别结果整合而成的文本进行关键词提取，在每个题目中提取出 10 个关键词 (代码见附件中的 keyfind.py)。

在得到每个题目的 10 个关键词之后，我们再将所得的关键词与题库中每个题目的原题进行对照，去除那些在原题中未出现的关键词，最终对于每个题目筛选得到不超过 5 个关键词的集合，即为集合 S_q(代码见附件中的 keyofQuestion.py)。此时，我们发现 372 题未得到关键词，因此我们对 372 题进行了特殊处理。经过对题目分析可知，372 题中所含的停用词较多，最终去除题目中的停用词后，发现题目中的剩余词数不超过 5 个，因此便将剩余的词均作为该题的关键词。

4.2 题目的分类

4.2.1 根据关键词的粗略分类

在4.1中我们已经根据 OCR 的识别结果和题目内容得到了每个题目所包含的关键词，据此我们可以将这 1000 个题目重新进行整理（代码见附件中的 sortkey.py），得到包含所有关键词的列表，以及每个关键词 k 对应的题目集合 S_k，其中：

$$S_k = \{q \mid k \in S_q\}$$

随后我们再对关键词进行筛选（同样在 sortkey.py 中完成）。将合 S_k 中包含的题目个数小于等于 10 的关键词从我们的列表中删除。这是因为我们假设如果仅仅有不超过 10 个题目包含关键词 k，那么 k 并不能够作为一个用以分类的关键词。例如，实际生活中，对于关键词 “椭”，根据经验包含 “椭” 的题目一般都与椭圆相关，也就是圆锥曲线类的题目，相应地，包含 “椭” 的题目总数也会有很多，因此我们认为 “椭” 可以作为一个好的分类标准。但比如 “$\otimes$” 在中学数学中，只会出现在有新定义特殊符号的题目中，比如 “路” 只会出现在一些应用题中，它们的出现频次都不会很高，也都不是很合适的分类标准。

在进行筛选后，我们发现剩余的所有 k 对应的 S_k 的并集中仅包括了 936 个题，这说明在筛选关键词之后有 64 个题目的所有关键词都被筛掉，我们再根据现有关键词进行分类的时候，分类的题目是不包含这 64 个题目的。这个结果也是我们可以接受的，因为这 64 个题目只与不超过 10 个题目共有某一个关键词，可以认为这 64 个题目题型特殊，题目所涉及的实际内容特殊，并不能够和其他的题目构成一类题。

4.2.2 根据关键词分类的整合

在4.2.1中我们得到的仅仅是每个关键词所对应的题目的集合，但这些关键词是可以整合的，即可能有多个关键词描述的是同一类题目，因此我们需要进行进一步的整合。(整合过程的代码见附件中的 sortkey.py)

首先，我们分别计算任意两个关键词 k_i 和 k_j 的关联程度 r_{ij}:

$$r_{ij} = r_{ji} = card(S_{k_i} \cap S_{k_j})$$

再将关联程度 $r_{ij} \geq x_m$ 的 k_i 和 k_j 视为是相关的关键词，即他们可能描述的是同一个题目类别。即有很多题目都同时包含这两个关键词时，我们可以认为包含这两个关键词中的任何一个的题目都属于同一类别。于是我们可以得到一个列表 relation_lst，在这个列表中储存了若干集合，每个集合均包含两个相关的关键词。

随后我们再对这个 relation_lst 进行扩充，如果有三个集合 S_1、S_2 和 S_3 满足:

$$(S_1 \cup S_2) \setminus (S_1 \cap S_2) = S_3$$

那么可以将 $S_1 \cup S_2 \cup S_3$ 加入 relation_lst。即 relation_lst 中的所有集合，均满足其元素(关键词)之间两两相关联。

进行第一次扩充之后都可能会出现包含三个元素的新集合，再将扩充后的 relation_lst 进行再次扩充可以得到更大的新集合。在 $x_m \in [3, 9]$ 的时候，将扩充过程迭代 5 次之后发现一定不再产生新的集合。

在完成扩充之后，我们再进行一次筛选，将 relation_lst 中包含元素小于等于 y_m 的集合删掉，并且将 relation_lst 中其他集合真包含的集合删掉。前者是把一些规模较小的类别删去，只保留规模较大的类别；后者是为了去除分类过程中出现的子类别，即免除一个大类别下的进一步划分。例如们在实际生活中会遇到立体几何题，在立体几何题中也会遇到包含三棱锥的题，我们的分类只需要划分到是否是立体几何题即可。

考虑到 x_m 与 y_m 的取值会直接影响我们分类的效果，我们记 $f(x_m, y_m)$ 为最终 relation_lst 中剩余的集合数量，$1000 - g(x_m, y_m)$ 为最终 relation_lst 中所有集合包含的所有关键字 k 对应的所有题目，即 $g(x_m, y_m)$ 为 relation_lst 中所有分类所不能包括的题目个数。根据4.2.1的分析，$g(x_m, y_m) \geq 64$。调整 x_m 与 y_m，我们得到 $f(x_m, y_m)$ 与 $g(x_m, y_m)$ 如表1所示，其中表格内容为 (f, g)。

我们希望 $f(x_m, y_m)$ 比较小，使得我们的分类可以不分得过于零碎，可以整体上分成数量并不多的几大类；我们希望 $g(x_m, y_m)$ 比较小，使得我们的分类可以涵盖 1000 个题目中尽可能多的题目，综合考虑我们选取参数 $(x_m, y_m) = (5, 2)$ 来进行分类整合。整合之后我们得到的 29 个类别分别为:

{128, 170, 18}, {18, 42, 31}, {56, 18, 27}, {18, 69, 30}, {18, 77, 30}, {108, 22, 39},{33, 22, 39}, {195, 29, 39}, {84, 37, 102}, {37, 102, 31}, {65, 22, 79}, {65, 174, 22}, {281, 131, 111}, {281, 98, 131}, {346, 91, 21}, {75, 356, 21}, {40, 19, 21}, {96, 21, 78}, {96, 236, 78}, {18, 20, 27, 31}, {20, 37, 27, 31}, {18, 66, 27, 31}, {384, 65, 22, 39}, {165, 86, 39, 22}, {19, 21, 75, 399}, {96, 21, 294, 40}, {49, 19, 22, 39, 29}, {64, 66, 740, 281, 31}, {19, 21, 91, 75, 95}

表 1: $f(x_m, y_m)$ 与 $g(x_m, y_m)$ 取值表

y_m \ x_m	3	4	5	6	7	8	9
1	(90, 84)	(65, 136)	(47, 175)	(40, 209)	(35, 251)	(27, 261)	(25, 303)
2	(70, 155)	(45, 208)	(29, 263)	(14,398)	(15, 400)	(12, 402)	(7, 488)
3	(42, 243)	(20, 381)	(10, 423)	(5, 477)	(4, 597)	(4, 597)	(3, 689)
4	(11, 452)	(4, 595)	(3, 597)	(3, 597)	(2, 692)	(1, 854)	(1, 854)

其中有 29 个集合，每个集合包含了几个关键词 k_i，作为一个对题目的分类。每个分类包含的题目为 S_{k_i} 的并集，即至少含有这个类别中一个关键词的所有题目。

4.2.3 分类的进一步调整

在4.2.2中完成分类整合之后我们仍需要对我们的分类进行进一步的调整。

我们需要对一部分集合做第一步整合。我们将有超过 3 个相同关键词的集合整合为一个大集合，即有超过三个相同关键词的类别看做是一类。这样做的依据是：整合之前我们的集合中的关键词总数不超过 5 个，如果有 3 个关键词相同，可以认为两个类别中有大约一半的题目是相同的，那么可以认为它们其实是同一类题。

根据我们的观察，整合了以下三处：

{18, 20, 27, 31}, {20, 37, 27, 31} $\longrightarrow$ {18, 20, 27, 31, 37}

{19, 21, 91, 75, 95},{19, 21, 75, 399} $\longrightarrow$ {19, 21, 91, 75, 95,399}

{18, 20, 27, 31, 37}, {18, 66, 27, 31} $\longrightarrow$ {18, 20, 27, 31, 37, 66}

整合之后共有 26 类，并将剩余未分类的题目统一归入 “others” 类别，对每一类包含的题目个数及题目序号进行统计，再统计每一个题目每天的总 PV 量 (详见 PVstatistics.py)，进而计算出这 27 个类别每天的总 PV 量 (详见 sortedPV.py)。

考虑到每个类别题目总数不同，我们为了纵向比较每个类别的查询热度，将每天的总 PV 量除以类别中题目的个数，得到类别中平均每题的日访问量。同时也考虑到每天的所有题目总 PV 量也有差异，实际上，14 天中大部分时候总 PV 量在 23 万附近，但第 4、5 天只有 15 万左右，第 6 天只有 17 万左右。我们猜测这可能与周末等周期性因素有关，为了横向比较每个类别每天相对查询热度的变化，我们将之前的结果再除以每天的所有题目总 PV 量，得到一天中某个类别平均每题被访问量占总访问量的比例（后称：访问率）。

将得到的 27×14 个数据绘制成图像，我们会发现有很多类别的变化趋势非常相似，甚至得到的曲线几乎重合，如图2所示。我们假设同一类别的题有相同的趋势，那么这些变化趋势很接近的类别也是可以合并成一类的。之所以会产生这种现象，是因为有的类别仍有 2 个共同关键词，使得这些类别包含的题目有很大一部分是重合的。

因此我们根据图像体现出来的结果，将原本的 26 个类别（不包括 “others” 类别）进一步进行合并，最终得到 13 个类别和一个 “others” 类别：

{128, 170, 18}, {18, 42, 31}, {56, 18, 27}, {18, 69, 77, 30}, {108, 33, 22, 39, 195, 29, 384, 65, 165, 86}, {84, 31, 37, 102}, {65, 22, 79,174}, {281, 131, 111,98}, {346, 91, 21,75,356,40,19,96,78,294,95,399}, {96, 236, 78},{18, 20, 27, 31, 37, 66}, {49, 19, 22, 39, 29}, {64, 66, 740, 281, 31}, “others”

前 13 个类别平均每个类别包含 126 个题目，可见不同类别之间也有相同的题目存在，在实际生活中，某一个题目也可能因为同时涵盖不同的知识点而处于不同的类别中。

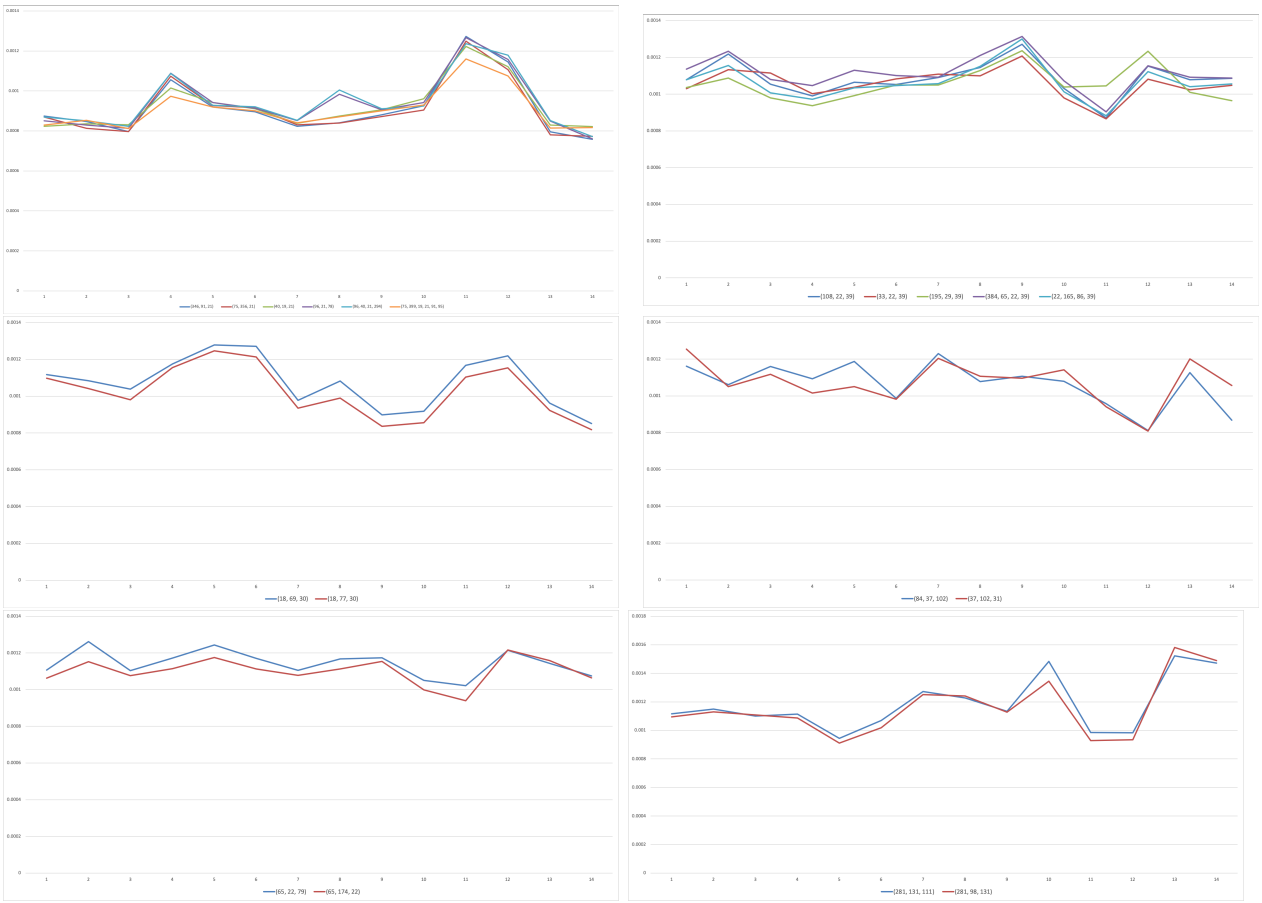

图 2: 部分可合并类别数据分布图

我们再对这 14 个类别按照前面的计算方法计算出每天每个类别的访问率，可以发现””others” 类的访问率在 14 天中近似不变，说明我们的分类是符合我们的期望的：除了 13 个大类以外其余的题目每天的访问率是差不多的，不会因为总体知识进度的改变而变化。

对于 13 个大类的访问率，我们在下面进行更详细的分析和预测。

4.3 不同类别的 PV 预测

在4.2中，我们已经得到了简化的关键词分类，之后便需要对每一个类别的 PV 值进行预测。在此，我们采用了一个 BP 神经网络模型来对数据进行预测 [3]。

4.3.1 BP 神经网络算法描述

神经网络是由输入层、隐含层和输出层组成，如图3所示。

对于神经网络的学习算法，具体可分为正向传播和反向传播。正向传播即为从输入层经过隐含层传至输出层，如果在输出层得到了预想的输出，则可以结束算法，否则便进行反向传播。反向传播便是按照通路反向计算，通过梯度下降法调整各个神经元之间的权值，使输出的误差减小。

具体的算法实现如下：

(1) 给定初始化的权值 $\theta = (\theta_{ij})$，使其为较小的随机非零值。

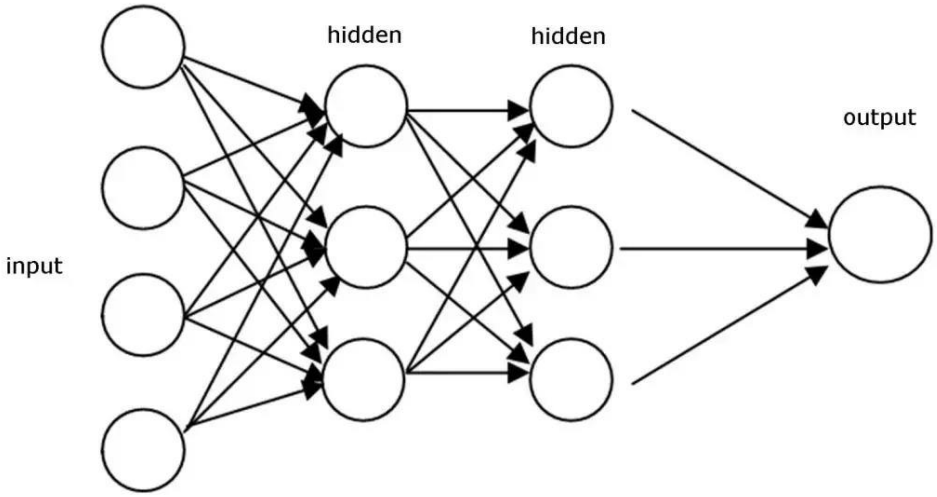

图 3: BP 神经网络结构图

(2) 给定神经网络的输入值 $X = (X_{ij})$，以及期望输出 $Y = (Y_i)$，对每一个样本进行如下 (3)∼(5) 的迭代，直到输出结果的误差小于阈值 ϵ。

(3) 从输入层开始计算隐含层及输出的状态 $Z^{(l+1)} = f(\theta^{(l)^T} Z^{(l)})$。

(4) 反向传播计算误差函数的梯度。对于输出层有 $\delta = (Z^{(l)} - Y).*f'(\theta^{(l-1)^T} Z^{(l-1)})$, 对于隐含层有 $\delta^{(l)} = \theta^{(l)^T}\delta^{(l+1)}.*f'(\theta^{(l)^T} Z^{(l)})$。

(5) 对权重进行梯度下降 $\theta^{(l)} = \theta^{(l)} - \alpha\delta^{(l+1)} Z^{(l)}$。

4.3.2 BP 神经网络模型的建立

对于本题目中的数据，我们采用三天的所有类别的访问率作为已知样本 X，取一个类别的后一天的搜索率为期望输出 Y，即为 $(X_{ij}, Y_{ij}$ 中 i 为日期，j 为类别序号)

$$X = \begin{pmatrix} X_{11} & X_{21} & \dots & X_{k1} \\ X_{12} & X_{22} & \dots & X_{k2} \\ \vdots & \vdots & & \vdots \\ X_{3l} & X_{4l} & \dots & X_{k+3,l} \end{pmatrix} \qquad Y = \begin{pmatrix} Y_{4i} & Y_{5i} & \dots & Y_{k+4,i} \end{pmatrix}$$

为了提高神经网络的训练效率，我们首先对数据进行标准化处理，有

$$X' = 2(X - max(X))/(max(X) - min(X)) - 1 \qquad Y' = 2(Y - max(Y))/(max(Y) - min(Y)) - 1$$

此时便得到所有样本中的数据均在 $(-1, 1)$ 中。在神经网络中，我们选取误差函数为均方误差函数，正则化后为

$$cost = \frac{1}{N}\sum_{i=1}^{N}(Y_i - Z_i)^2 + \frac{\lambda}{2N}\sum_{i,j}\theta_{ij}^2$$

神经网络中的传递函数采用正切 Sigmoid 函数，最大迭代次数为 2000 次，以此神经网络来进行样本的训练和预测 (具体代码见附件)。之后，给出三天的所有类别的访问率作为输入，即可预测出后一天的所有类别各自的访问率。

4.4 热门题目的预测

根据4.3中的 BP 神经网络模型，用样本数据训练之后，输入连续三天各类别的访问率，就可以得到下一天每个类别的访问率。

我们再根据访问率从高到低排序，取前三个类别，记为当天的热门类别，可能当天学校的学习进度进行到了这三个类别相关的知识点。接下来我们就应该根据热门类别来预测热门题目。首先我们根据每个类别中所有题目在 14 天的总 PV 量来得到每个类别中的 Top 10 热门题目，例如：

{18, 42, 31}: (1, 21835), (11, 15298), (12, 14898), (13, 13923), (14, 13333), (17, 12735), (19, 11764), (28, 9773), (29, 9696), (30, 9668)

{281, 98, 131, 111}: (12, 14898), (13, 13923), (17, 12735), (26, 9875), (33, 9446), (53, 7294), (67, 6194), (71, 5992), (73, 5899), (96, 5239)

{39, 49, 19, 22, 29}: (3, 21095), (9, 16377), (25, 10176), (27, 9783), (40, 8547), (41, 8265), (60, 6590), (62, 6362), (63, 6333), (69, 6127)

其中 (1,21835) 代表题目 1 的总访问量为 21835.

根据之前得到的前三访问率类别，访问率最高的类别中取访问量前五的题目，访问量第二的类别取没有取过的前三访问量题目，访问量第三的类别取没有取过的前二访问量题目。

将得到的 10 个题目按照总访问量排序，便完成了对下一天 PV 量 Top 10 的题目的预测。

5 模型的检验

我们用已有数据中的前 13 天的数据对神经网络进行训练，之后对第 14 天的数据进行预测，可以得到第 14 天的访问率的数据，如表2所示

表 2: 第 14 天搜索率预测数值与实际数值对照表

类别序号	1	2	3	4	5	6	7
预测值	0.0009175	0.0012006	0.0009214	0.0009205	0.0010855	0.0009950	0.0010819
实际值	0.0009426	0.0010725	0.0008863	0.0008295	0.0009894	0.0010462	0.0010504
类别序号	8	9	10	11	12	13	
预测值	0.0013948	0.0009535	0.0009635	0.0011150	0.0011441	0.0010975	
实际值	0.0013925	0.0008095	0.000670	0.0010043	0.0010034	0.0009722	

利用预测的访问率，我们根据4.4中的预测方法，可以预测出第 14 天最热门的前 10 道题目的 ID 为 3,17,11,1,12,9,13,14,33,26。之后，我们将第 14 天实际的访问量的数据进行排序，得到预测题目在当天实际的排名，如表3所示

表 3: 预测热门题目的实际排名

题目 ID	3	17	11	1	12	9	13	14	33	26
实际排名	3	4	5	7	9	14	27	29	73	872

由此可以看出，我们预测的访问量前 10 名的题目中，有 5 个实际就处于前 10 名，同时还有 3 个处于 10~30 名，可见我们的模型预测的结果与实际的情况符合的较好。

6 模型的优缺点

6.1 模型的优点

1、本模型对题目根据关键词进行了科学的分类，较好的反映出了不同题目之间的关联。

2、模型中采用 BP 神经网络进行预测，能够科学的对于题目的访问率进行预测。

3、本模型在处理过程中主要考虑访问量整体的趋势，忽略了一些细微的差别，具有一定的鲁棒性。

6.2 模型的缺点和改进

1、由于本题中所给的 OCR 识别结果只有数字，因此我们使用 jieba 库处理的过程中，只能够对单个数字 (即单字或单个特殊符号) 进行提取，从而无法有效的去除多个字组成的停用词。如果能够获得题目的汉字原文，在利用 jieba 库进行处理时，便可以比较好的处理多个字组成的停用词，从而使关键词提取更加准确。

2、模型中采用的 BP 神经网络的算法在样本数据量较大时效果比较好，而在本题中只有 14 天的数据，因此在数据量较小的情况下，神经网络模型的误差可能会较大。如果能够利用更多日期的数据对神经网络进行训练，那么就会使关键词访问率的预测更加准确。

参考文献

[1] fxlou 的 CSDN 博客. prophet 模型小结. https://blog.csdn.net/fxlou/article/details/79576493. (2020/05/06)

[2] Rob J Hyndman, George Athanasopoulos. Forecasting: Principles and Practice. https://otexts.com/fpp2/. (2020/05/06)

[3] 杨雪梅, 郭家勇. 基于 BP 神经网络对网站访问量的预测 [J]. 连云港职业技术学院学报, 2009, 022(003):5-7.