第九节课 LSH 近似最近邻查找与YouTube推荐系统

https://github.com/cystanford 老师的GitHub
YouTube论文真的有太多可以学习的东西了！！！

第九节课笔记

讲义内容概括	知识点	Question?

最近邻查找

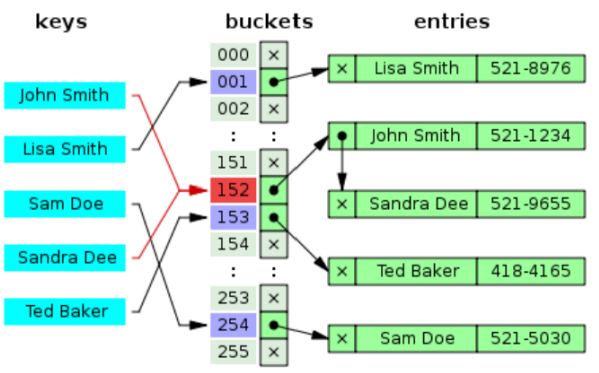
- NN，Nearest Neighbor Search，最近邻查找问题
- KNN，K-Nearest Neighbor，k最近邻，查找离目标数据最近的前k个数据项
- ANN，Approximate Nearest Neighbor，近似最近邻检索，在牺牲可接受范围内的精度的情况下提高检索效率
- 最近邻检索是线性复杂度的，当处理大规模数据时可以采用ANN方法
- LSH，局部敏感哈希是ANN的一种LSH-local sensitive Hash

索引技术：

- 基于树的索引技术（二叉树，B-Tree,B+Tree）
- 基于哈希的索引技术
- 基于词的倒排索引?
- 海量数据的检索方式，Hash是重要的索引技术

B树中的B指的是balance，平衡二叉树
MySQL使用的是B+Tree (读作B加树)
Oracle 使用的是B-Tree（读作B树)
Postgre 开源免费版本，分析型数据库效率高，可以存储TB以上量级

Hash 通过keys在buckets里面查找，如果两个keys指向同一bucket，那么再通过链表查找



LSH算法

传统的HashTable用于检索数据，无法将相似的数据放到同一个Bucket中，比如h=x mod wLSH将相邻的数据，通过映射后依然保持相邻的关系，即保持局部的敏感度Locality-Sensitive

LSH，Locality-Sensitive Hashing，局部敏感哈希需要查找与某个数据1个或多个相似的数据

最近邻查找方法（ANN，Approximate Nearest Neighbor）

MinHash算法原理-适合短文本

文档相似度计算

将原来的Jaccard相似度计算，等同于降维后的相似度矩阵计算（Input Matrix => Signature Matrix）原来文档的Jaccard相似度高，那么它们的hash值相同的概率高

MinHash：

- 特征矩阵按行进行随机排列后，第一个列值为1的行的行号
- Thinking 最小哈希值=?

h(C1)=2，h(C2)=1，h(C3)=1，h(C4)=3

元素	C1	C2	C3	C4
定制	0	1	1	0
AI	1	0	1	0
芯片	0	1	0	1
成为	1	0	1	1
趋势	1	0	1	1

Thinking Ci与Cj的MinHash相等的概率P(h(Ci)=h(Cj))，与Ci,Cj Jaccard相似度的关系相等:

对于同一行，有可能都为1（情况A，出现a次），有可能都为0（这种没有计算的意义），可能一个维1一个为0（情况B，出现b次）

Jaccard = a/(a+b)

P(h(Ci)=h(Cj)) (h(Ci)=h(Cj))发生的概率是第一行都是1比上第一行可能为1可能为0，即第一行出现A和B的概率，=a/(a+b)

P(h(Ci)=h(Cj))= sim(Ci,Cj)用Ci，Cj的MinHash值相等的概率，对他们的Jaccard相似度进行估计

MinHash：

1. 分别经过3次随机置换（红、黄、蓝）
2. 每次置换后，采用MinHash得到Signature
3. 使用Sig矩阵相似度用来近似估计原始矩阵Input Matrix的Jaccard相似度

打擂法：
如何对海量数据进行排序
=> 存储空间，计算时间
=> 有多个hash函数,通过hash i得到最新的行号，如果此时列上的元素为1，而且新行号比原来记录的M值小，那么更新M值

MinHash执行：

对于hash函数：

h(x)= x mod 5

g(x)=(2x+1) mod 5

- 行数+1，每次进行h和g函数运算
- 如果列上的值为1，那么查看Hash得到的数值是否更小，更小则更新

=> 通过m个针对row index的Hash函数，完成m次行向量的置换（解决了行向量置换的问题）

C1	C2
1	0
0	1
1	1
1	0
0	1

Hash函数	C1	C2
h(1)=1	1	无穷大
g(1)=3	3	无穷大
h(2)=2	1	2
g(2)=0	3	0
h(3)=3	1	2
g(3)=2	2	0
h(4)=4	1	2
g(4)=4	2	0
h(5)=0	1	0
g(5)=1	2	0

Mini-Hash降维，将原本计算Jaccard所需的len(vec)(例如1000）降维成n个（例如32个）这里的n体现在mod n

LSH是相似度的近似求解方式
在MinHash基础上，将Signature向量分成多段(band)
将Signiture矩阵分成b组，每组由r行组成对每一组进行hash，各个组设置不同的桶空间。只要两列有一组的MinHash相同，那么这两列就会hash到同一个桶而成为候选相似项。
Locality-Sensitive Hashing是满足一定条件的Hash函数簇
令d1<d2是定义在距离测定d下得两个距离值，
如果一个函数族的每一个函数f满足：
如果d(x,y)<=d1,则f(x)=f(y)的概率至少为p1，即P(f(x)=f(y)) >= p1
如果d(x,y)>=d2,则f(x)=f(y)的概率至多为p2，即p(f(x)=f(y)) <= p2

那么称F为(d1,d2,p1,p2)-sensitive的函数族。
Jaccard相似性对应的LSH为MinHash,是(d1,d2,1-d1,1-d2)-sensitive

datasketch
datasketch中的MinHash()：

- num_perm参数，Hash置换函数设定个数，默认为128，如果需要提高精度，可以提高该数值，比如设置num_perm=256
- update函数，内容Hash化 m1.update(content)
- merge函数，Hash合并，比如 m1.merge(m2)

datasketch中的MinHashLSH()：
http://ekzhu.com/datasketch/lsh.html

- threshold 参数，Jaccard 距离阈值设定，默认为0.9
- num_perm参数，Hash置换函数设定个数，默认为128
- weights (tuple, optional)，优化Jaccard 阈值，能够弹性选择
- params (tuple, optional)，bands 的数量与规模大小
- insert(key)，内容载入LSH系统
- remove(key)，移除相关hash值
- query(key)，查询内容需要时minHash化

datasketch中的MinHashLSHForest()：
局部敏感随机投影森林
http://ekzhu.com/datasketch/lshforest.html 论文：
http://ilpubs.stanford.edu:8090/678/1/2005-14.pdf
求近似最近邻方法的一种（ANN）
随机选取一个从原点出发的向量，与这个向量垂直的直线将平面内的点划分为了两部分。
当数目比较大的时候，可以继续进行划分 对应于一棵深度为2，有4个叶节点的树（划分出4个部分）。
一直划分，直到每个叶节点中点的数目都达到一个足够小的数目，也就是将每次搜索与计算的点

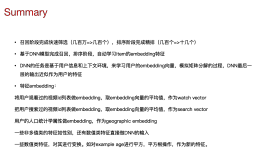
LSH 近似最近邻查找

- 什么是近似最近邻查找
- 什么是Hash
- MinHash与Jaccard相似度
- MinHash的计算
- LSH算法
- SimHash算法
- 工具：datasketch
- 使用MinHashLSH对文本进行近似近邻查找
- 使用SimHash计

算文本距离

YouTube推荐系统

- 推荐系统的架构（基于DNN）
- 主要特征处理
- Example Age
- 样本和上下文选择
- 不对称的共同浏览问题
- 负采样
- 不同网络深度和特征的实验
- 排序阶段的建模
- 排序阶段的特征工程（分类特征、连续特征）
- 隐藏层的实验

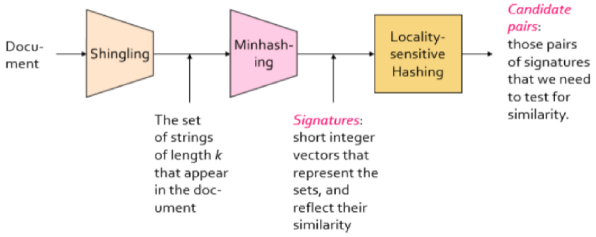


的数目减小到一个可接受的范围。
建立多个随机投影树构成随机投影森林，将森林的综合结果作为最终的结果

- num_perm参数，Hash置换函数设定个数，默认为128
- l 参数，代表prefix trees的数量，默认为8
- add(key)，内容载入LSHForest系统
- query(key, k)，查询与key相似的Top-K个邻居

datasketch中的MinHashLSHEnsemble()：
http://ekzhu.com/datasketch/lshensemble.html
论文：
http://www.vldb.org/pvldb/vol9/p1185-zhu.pdf
对于相似度的另一种计算方式，对于X和X'，Jaccard计算差很大

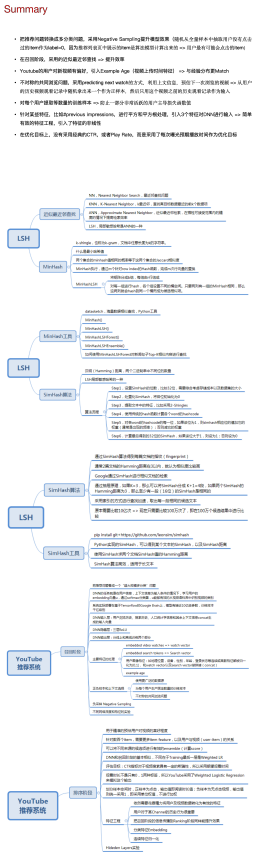
- threshold 参数，Jaccard 距离阈值设定，默认为0.9
- num_perm参数，Hash置换函数设定个数，默认为128
- index()，内容载入LSHEnsemble系统 **每次建树要使用一次
- query(key, size)，查询与key相似的邻居



Step1，对文档进行k-shingle，即将文档切割成一个一个的元素，这些元素是由很多的字符串组成的（由k个字符串组成）
Step2，使用MinHash得到集合元素的签名
Step3，使用LSH加快候选相似对的查找，得到可能的候选对

数据集：天猫双11新闻
对新闻中的句子进行检索使用MinHashLSHForest，针对某句话Query进行

1. 基于词的倒排索引?
2. 密码学？MD5? hash
3. LSH与apporior相似和差别
4. SIMHash Step4，使用传统的hash函数计算各个word的hashcode比如："th".hash = -502157718，"he".hash = -369049682，？？Step5，对各word的hashcode的每一位，如果该位为1？权重是出现的频率吗?
5. SimHash算法：--为什么可以区别?每个单词（特征）的01表示和权重的乘积再求和再转换为01，不相似的文章也有可能fingerprint相似吧
6. softmax分类器
7.



TOP-K相似度输出

Step1、分词

Step2、创建MinHash

Step3、使用MinHashLSHForest进行Index

Step4、使用MinHashLSHForest进行Query，查询Top-K相似句子

SimHash算法--适合长文本

汉明（Hamming）距离：两个二进制串中不同位的数量，汉明距离在0-10之间认为是相似的

LSH局部敏感哈希的一种Paper: similarity estimation techniques from rounding algorithms

Google采用SimHash进行网页查重 Detecting Near-Duplicates for Web Crawling

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.7794&rep=rep1&type=pdf

SimHash算法：--为什么可以区别?

Step1，设置SimHash的位数，比如32位，需要综合考虑存储成本以及数据集的大小

Step2，初始化SimHash，将各位初始化为0

Step3，提取文本中的特征，比如采用2-Shingles "the cat sat on the mat"=>{"th", "he", "e ", " c", "ca", "at", "t ", " s", "sa", " o", "on", "n ", " t", " m", "ma"}

Step4，使用传统的hash函数计算各个word的hashcode 比如："th".hash = -502157718，"he".hash = -369049682，……

Step5，对各word的hashcode的每一位，如果该位为1，则simhash相应位的值加它的权重（通常是出现的频率）；否则减它的权重

Step6，累加计算最后得到的32位的SimHash，如果该位大于1，则设为1；否则设为0

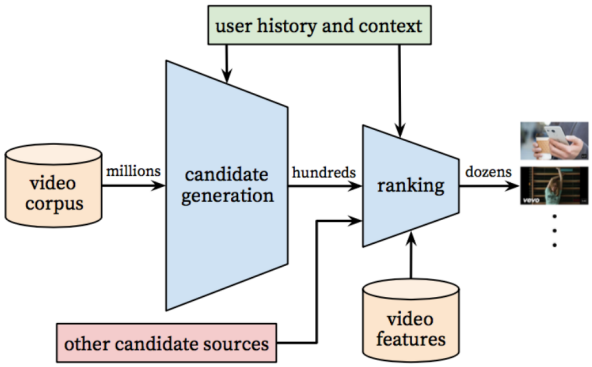
在文本量巨大的情况下：

如果SimHash有64位，Hamming距离<=3认为相似，相似的SimHash有C(64,3)种可能

通过抽屉原理，如果K=3，那么可以将SimHash分成 K+1=4段，如果两个SimHash的Hamming距离为3，那么至少有一段（16位）的SimHash是相同的，分成4段，分别查找，只要有一段匹配就是候选采用数据库索引的方式进行匹配查找效率高—先利用抽屉原理filter，使得查找数据从2^34变成2^(34-16)，再使用Haming距离

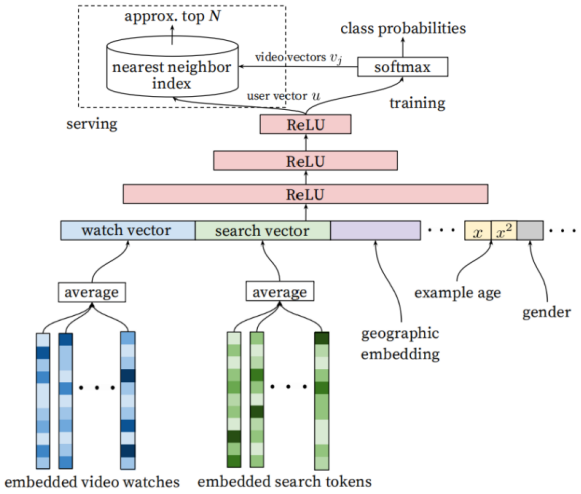
SimHash：pip install git+https://github.com/leonsim/simhash
分词影响

YouTube推荐系统



Deep Neural Networks for YouTube Recommendations, 2016
https://dl.acm.org/citation.cfm?doid=2959100.2959190
提出推荐系统分为召回（候选集生成）和排序两个阶段

- 召回阶段，基于用户画像及场景数据从海量的视频库（百万级别）中将相关度最高的资源检索出来，作为候选集 —DNN 3层塔型—经典结构 “超大规模多分类”问题 softmax
- 召回阶段可以通过“粗糙”的方式召回候选item
- 排序阶段，基于更加精细的特征对候选集（百级别）进行排序，最终呈现给用户的是很少的一部分数据。
- Ranking阶段，采用更精细的特征计算user-item之间的排序score，作为最终输出推荐结果的依据



将用户观看历史和搜索历史通过embedding的方式映射成为一个稠密的向量，同时用户场景信息以及用户画像信息（比如年龄，性别等离散特征）也被归一化到[0,1]作为DNN的输入——统一的向量

主要特征的处理：

- embedded video watches => watch vector，用户的历史观看是一个稀疏的，变长的视频id序列，采用类似于word2vec的做法，每个视频都会被embedding到固定维度的向量中。最终通过加权平均（可根据重要性和时间进行加权）得到固定维度的watch vector
- embedded search tokens => Search vector，和watch vector生成方式类似
- 用户画像特征：如地理位置，设备，性别，年龄，登录状态等连续或离散特征都被归一化为[0,1]， 和watch vector以及search vector做拼接（concatenate）
- 推荐系统中的example age（样本年龄）：用户更倾向于推荐尽管相关度不高但是新鲜fresh的视频。 Example age特征表示视频被上传之后的时间。

正负样本和上下文选择：
在有监督学习问题中，label选择非常关键，因为label决定了你做什么，决定了你的上限，而feature和model都是在逼近label
使用更广泛的数据源，训练样本要用youtube上

的所有视频观看记录，而不只是系统推荐的视频观看记录。否则，面对新视频的时候很难推荐，并且推荐器会过度偏向exploitation
为每个用户生产固定数量的训练样本，在损失函数中所有用户的权重一样 => 防止一部分非常活跃的用户主导损失函数值

样本和上下文选择中的不对称的共同浏览问题（asymmetric co-watch）

负采样 Negative Sampling数据处理过程有时候比算法更重要（
采用负采样，也就是随机从全量item中抽取用户没有点击过的item作为label=0的item后，效果明显提升。

YouTube的排序阶段：
相比召回阶段，引入了更多的feature（当前要计算的video的embedding，用户观看过的最后N个视频embedding的average，用户语言的embedding和当前视频语言的embedding，自上次观看同channel视频的时间，该视频已经被曝光给该用户的次数）

排序阶段的建模（对观看时间）：
排序阶段中的特征工程（Feature Engineering）：
排序阶段中的分类特征Embedding（Embedding Categorical Features）：
排序阶段中的连续特征归一化（Normalizing Continuous Features）：

总结Summary

- 记录作业内容
  - Thinking1：什么是近似最近邻查找，常用的方法有哪些
  - Thinking2：为什么两个集合的minhash值相同的概率等于这两个集合的Jaccard相似度
  - Thinking3：SimHash在计算文档相似度的作用是怎样的?
  - Thinking4：为什么YouTube采用期望观看时间作为评估指标
  - Thinking5：为什么YouTube在排序阶段没有采用经典的LR（逻辑回归）当作输出

层，而是采用了Weighted Logistic Regression?
    - Action1：使用MinHashLSHForest对微博新闻句子进行检索 weibo.txt针对某句话进行Query，查找Top-3相似的句子
    - Action2：请设计一个基于DNN模型的推荐系统阐述两阶段的架构（召回、排序）以及每个阶段的DNN模型设计： DNN输入层（如何进行特征选择） DNN隐藏层结构 DNN输出层

-

Codewar, leetcode，浙江大学ACM网站, pku
https://zoj.pintia.cn/problem-sets/91827364500/problems/91827364500

A+B
ProblemAccepted
Wrong Answer
Compilation Error
Non Zero Exit Error
Time Limit Exceeded
Memory Limit Exceeded
Segmentation Fault
Runtime Error

L9优秀作业
陈学良
https://github.com/xueliang787/RS06-09
陶学节
https://github.com/arthurt53/RS6-work/tree/master/L9