RS&NLP选修 Lesson-02

前情提要

- 根据业务场景不同，推荐系统可以演化得差异很大。因此，要注意学习算法背后的思想，做到融会贯通

- 实例：视频相关推荐。拆解和抽象为机器学习问题，进而用SGNS的方式去解决

- NLP和RS是算法的两个重要应用方向，其背后的思想可以找到很多相通和借鉴之处

- 理论提升：和矩阵分解的等价性

回顾：视频相关推荐

- 分为3个步骤：
  - Step1 构建序列：预处理用户的行为日志，使之适配算法
  - Step2 求解相似性：利用skip-gram with negative sampling计算item之间的相似性
  - Step3 Faiss建立索引：利用索引库为item建立索引，供线上推荐使用

了解你的数据

```
UserId,ProductId,Rating,Timestamp
A39HTATAQ9V7YF,0205616461,5.0,1369699200
A3JM6GV9MNOF9X,0558925278,3.0,1355443200
A1Z513UWSAAO0F,0558925278,5.0,1404691200
A1WMRR494NWEWV,0733001998,4.0,1382572800
A3IAAVS479H7M7,0737104473,1.0,1274227200
AKJHHD5VEH7VG,0762451459,5.0,1404518400
A1BG8QW55XHN6U,1304139212,5.0,1371945600
A22VW0P4VZHDE3,1304139220,5.0,1373068800
A3V3RE4132GKRO,130414089X,5.0,1401840000
A327B0I7CYTEJC,130414643X,4.0,1389052800
```

Description

This dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.

This dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

http://jmcauley.ucsd.edu/data/amazon/

Step1 构建序列

- Input：曝光/点击日志

```
UserId,ProductId,Rating,Timestamp
A39HTATAQ9V7YF,0205616461,5.0,1369699200
A3JM6GV9MNOF9X,0558925278,3.0,1355443200
```

- Output：点击序列

```
B004APTUFM      B004APYRKU      B00395KAPS      B004B4AWH2
B000I1OYNK      B001EJOPTS      B001G7PIJ4      B004NH138O      B00652DGSK
```

- Code：实操。一行语句生成训练语料。

- cat ratings_Beauty.csv | grep -v UserId | sort -t',' -nk4 | awk 'BEGIN{FS=",";OFS="\t"}$3>2{d[$1]=d[$1]$2";"}END{for(i in d) print i"\t"substr(d[i],0,length(d[i])-1)}' | awk '{split($2,a,";");if(length(a)>3)print $2}' | awk 'gsub(";","\t",$0)' > corpus.dat

用Linux shell 命令预处理 代码解读

1. cat ratings_Beauty.csv　//按行读取文件
2. grep -v UserId　//去掉标题行
3. sort -t',' -nk4　//按时间戳排序
4. awk 'BEGIN{FS=",";OFS="\t"}$3>2{d[$1]=d[$1]$2";"}END{for(i in d) print i"\t"substr(d[i],0,length(d[i])-1)}'　//按UserId聚合
5. awk '{split($2,a,";");if(length(a)>3)print $2}'　//过滤掉短session
6. awk ‘gsub(“;”,”\t",$0)'　//替换分隔符
7. > corpus.dat　//输出到文件

Step2 求解相似性

- Input：点击序列

```
B004APTUFM      B004APYRKU      B00395KAPS      B004B4AWH2
B000I1OYNK      B001EJOPTS      B001G7PIJ4      B004NH138O      B00652DGSK
```

- Output：item -> vector

```
B004OHQR1Q -0.14790097 -0.88445485 1.7531323 1.3609
B000ZMBSPE 0.5054478 -0.26883852 -0.26304248 0.2367
```

- Code：实操。

```
v_size = 8
v_window = 3
v_min_count = 2
v_workers = 2

corpusFilePath = './data/corpus.dat'

corpusFile = open(corpusFilePath, u'r')
model = Word2Vec(
    LineSentence(corpusFile),
    size=v_size,
    window=v_window,
    min_count=v_min_count, workers=v_workers)
```

Step3-1 建立索引

- Input：item -> vector
- Output：索引文件
- Code：实操

Step3-2 用索引查找

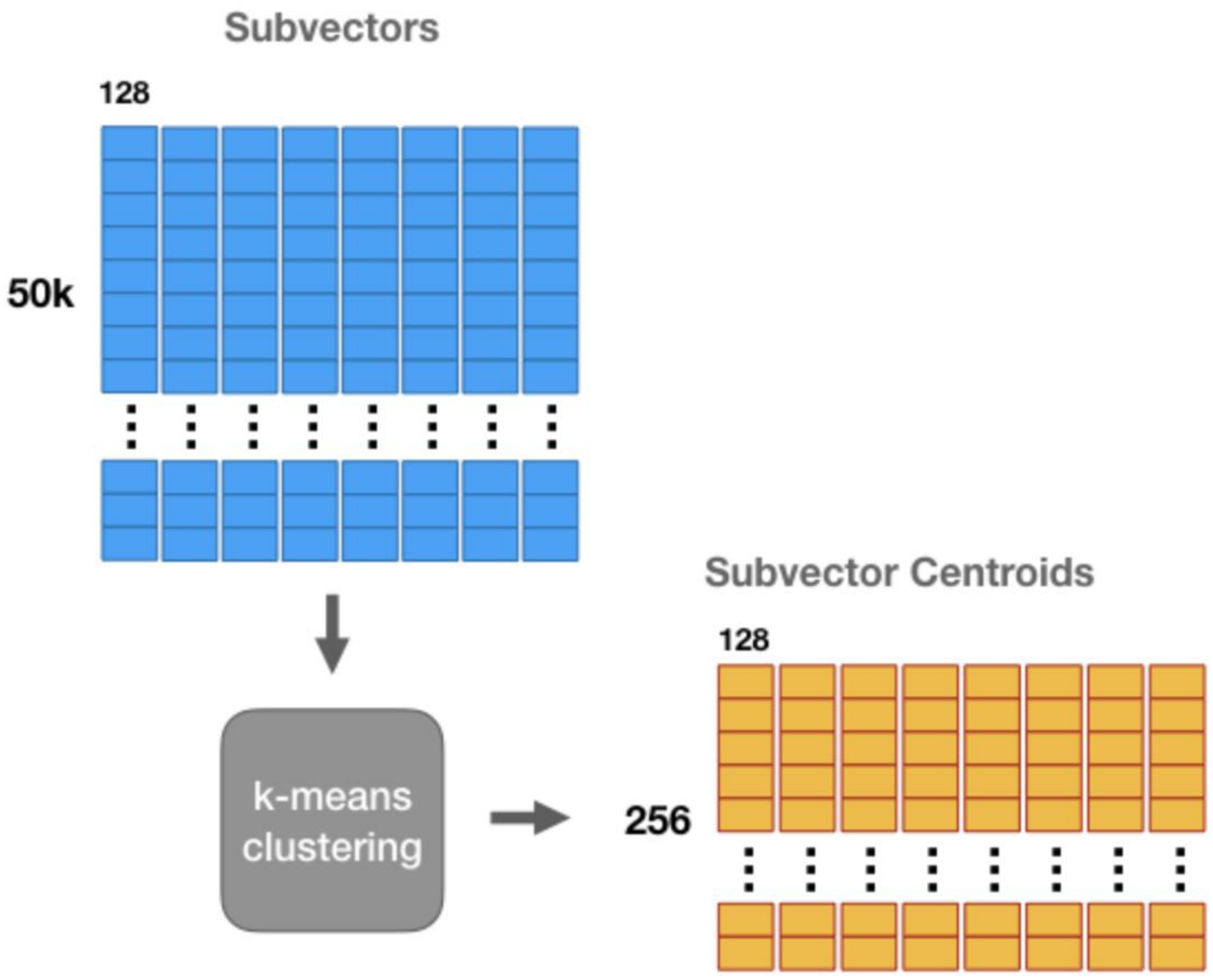
- Input：索引文件，key
- Output：key的最近邻
- Code：实操

Index factory

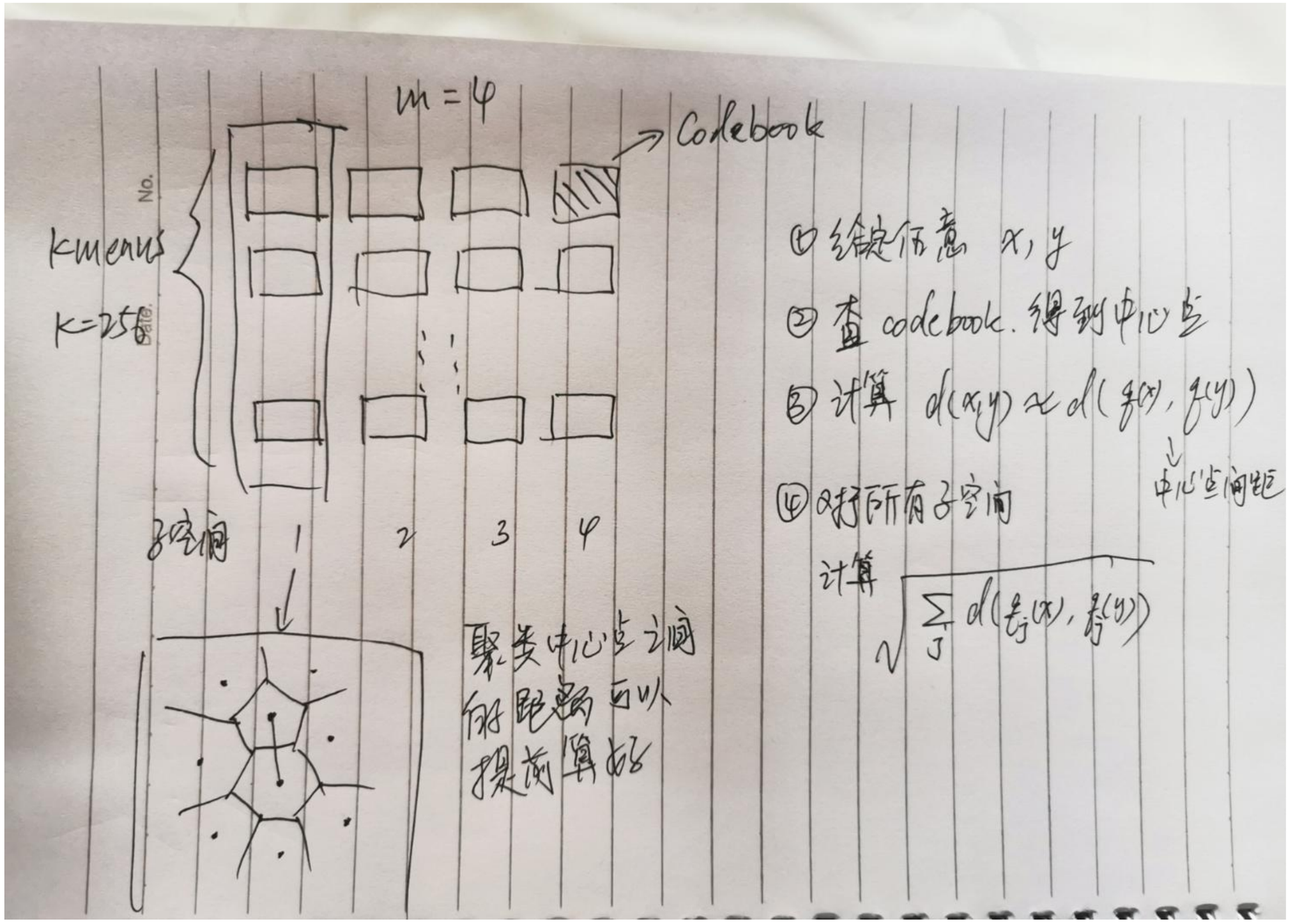
Method	Class name	index_factory	Main parameters	Bytes/vector	Exhaustive	Comments
Exact Search for L2	IndexFlatL2	Flat	d	4*d	yes	brute-force
Exact Search for Inner Product	IndexFlatIP	Flat	d	4*d	yes	also for cosine (normalize vectors beforehand)
Hierarchical Navigable Small World graph exploration	IndexHNSWFlat	'HNSWx,Flat`	d, M	4*d + 8 * M	no	
Inverted file with exact post-verification	IndexIVFFlat	IVFx,Flat	quantizer, d, nlists, metric	4*d	no	Take another index to assign vectors to inverted lists
Locality-Sensitive Hashing (binary flat index)	IndexLSH	-	d, nbits	nbits/8	yes	optimized by using random rotation instead of random projections
Scalar quantizer (SQ) in flat mode	IndexScalarQuantizer	SQ8	d	d	yes	4 bit per component is also implemented, but the impact on accuracy may be inacceptable
Product quantizer (PQ) in flat mode	IndexPQ	PQx	d, M, nbits	M (if nbits=8)	yes	
IVF and scalar quantizer	IndexIVFScalarQuantizer	IVFx,SQ4 "IVFx,SQ8"	quantizer, d, nlists, qtype	SQfp16: 2 * d, SQ8: d or SQ4: d/2	no	there are 2 encodings: 4 bit per dimension and 8 bit per dimension
IVFADC (coarse quantizer+PQ on residuals)	IndexIVFPQ	IVFx,PQy	quantizer, d, nlists, M, nbits	M+4 or M+8	no	the memory cost depends on the data type used to represent ids (int or long), currently supports only nbits <= 8
IVFADC+R (same as IVFADC with re-ranking based on codes)	IndexIVFPQR	IVFx,PQy+z	quantizer, d, nlists, M, nbits, M_refine, nbits_refine	M+M_refine+4 or M+M_refine+8	no	

根据需求去选择
具体算法

PQ算法

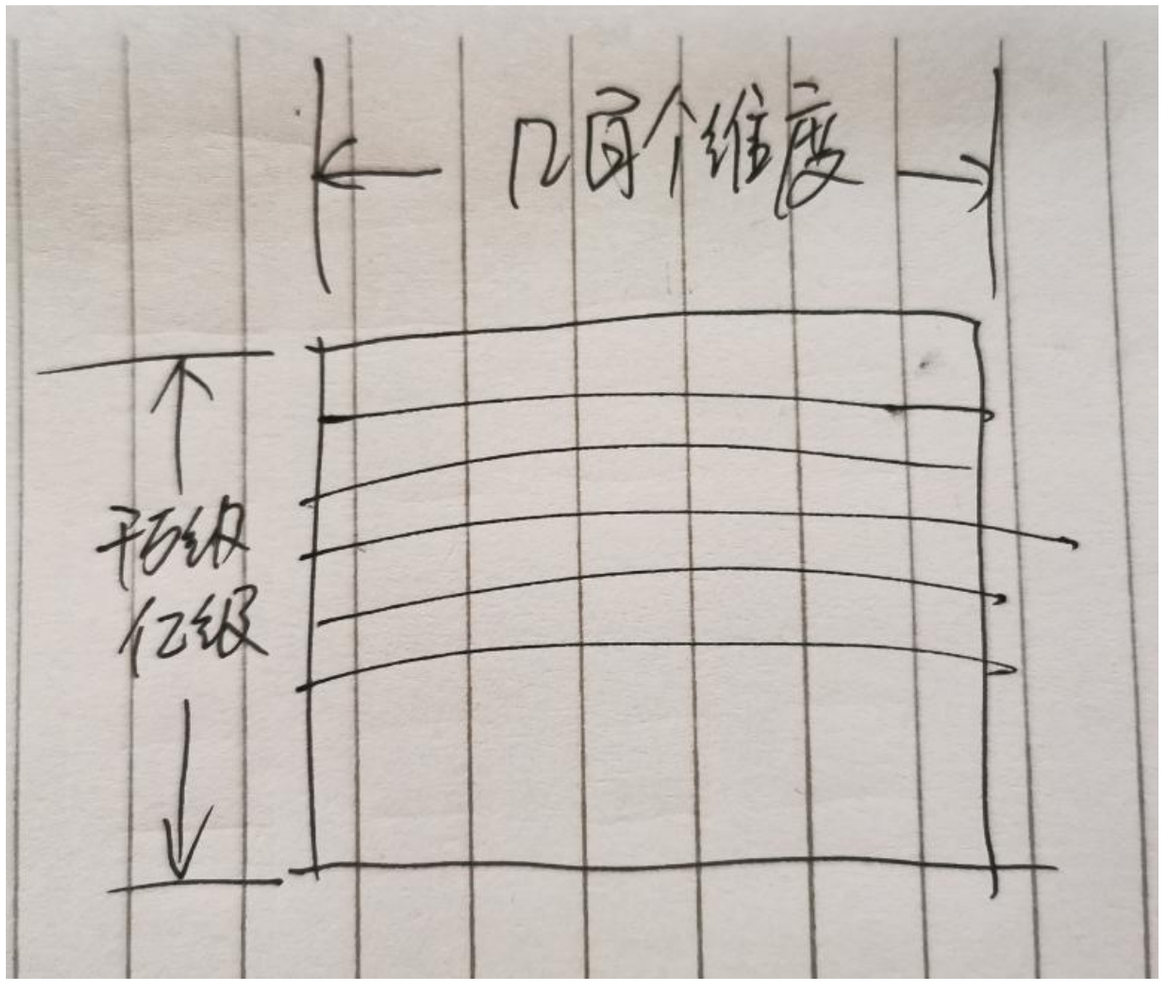

Subvectors
128
50k
k-means
clustering
Subvector Centroids
128
256

算法解析


m=4
Codebook
No.
kmeans
K=256
Date.
子空间
1
2
3
4
聚类中心点之间
的距离可以
提前算好
① 给定任意 x, y
② 查 codebook. 得到中心点
③ 计算 d(x,y) ≈ d(f(x), f(y))
中心点间距
④ 对所有子空间
计算
$\sqrt{\sum_j d(q_j(x), q_j(y))}$

可行性分析

- 通常向量维度不太大，瓶颈在样本点的量级
- 面对千万级甚至亿级的样本点，运行kmeans不太适合
- 有一些改进的算法比如IVFADC，但计算量仍然很大

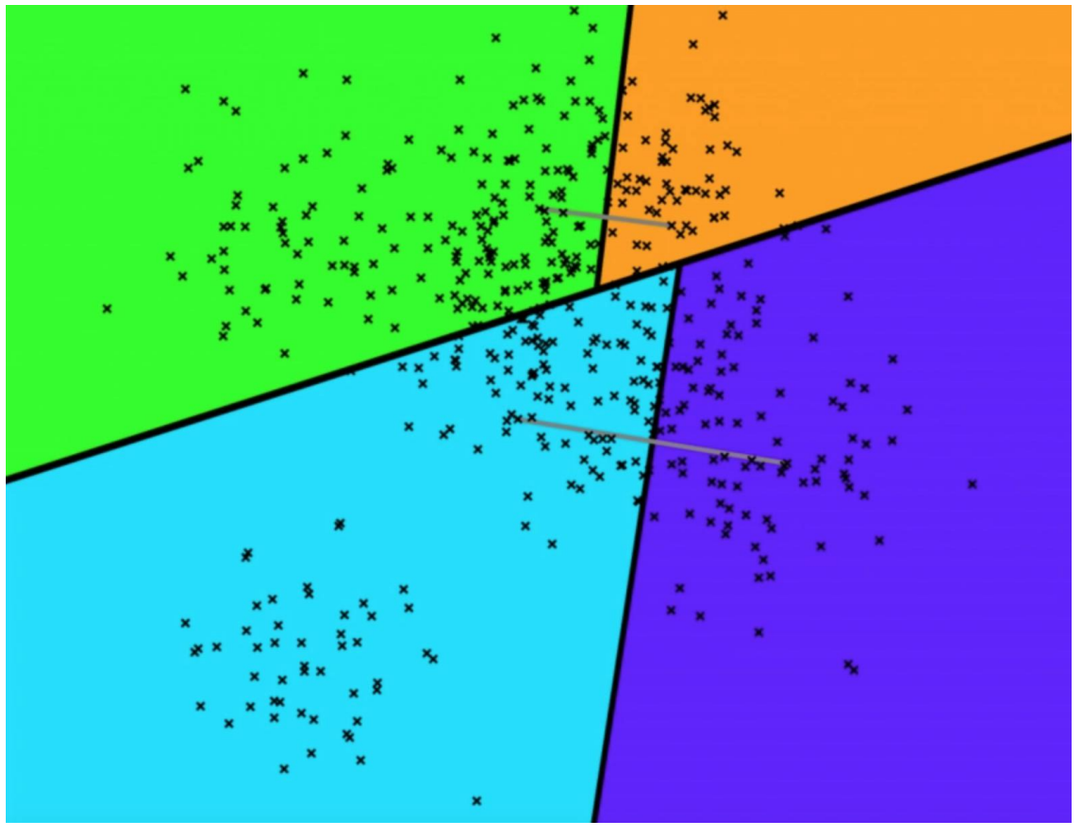


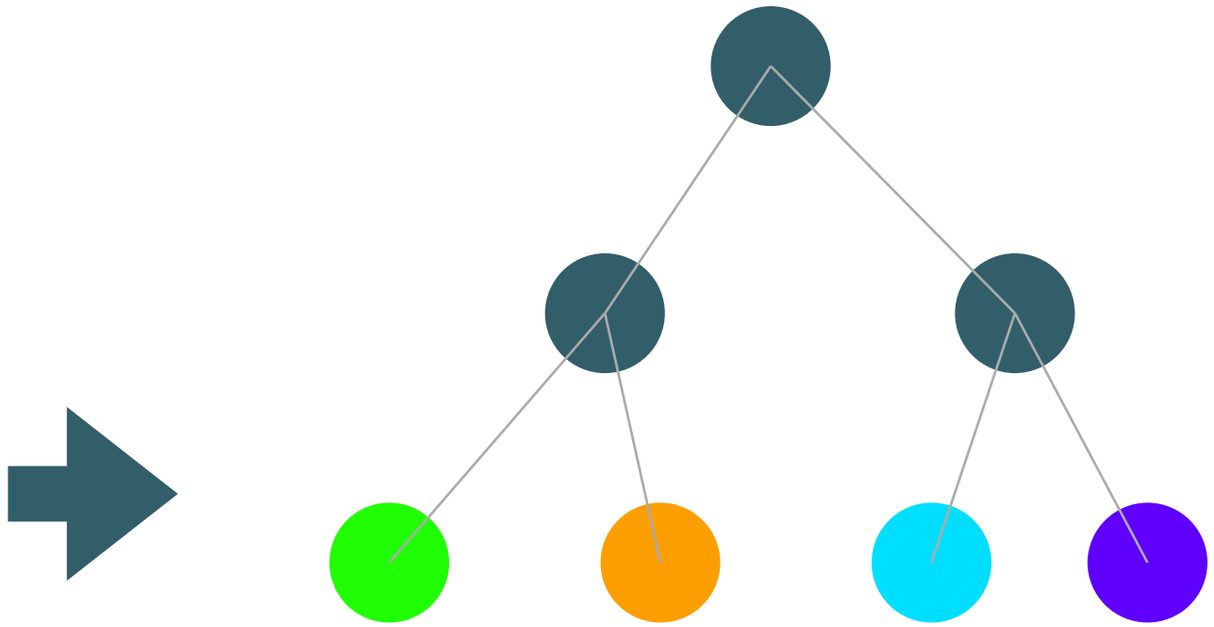
Annoy算法

- 利用二叉树的结构对空间进行随机划分，建索引阶段效率有所提升
- 二叉树结构在检索时效率也很高
- 构建多棵树形成森林，提高检索的召回率

https://github.com/spotify/annoy

算法解析

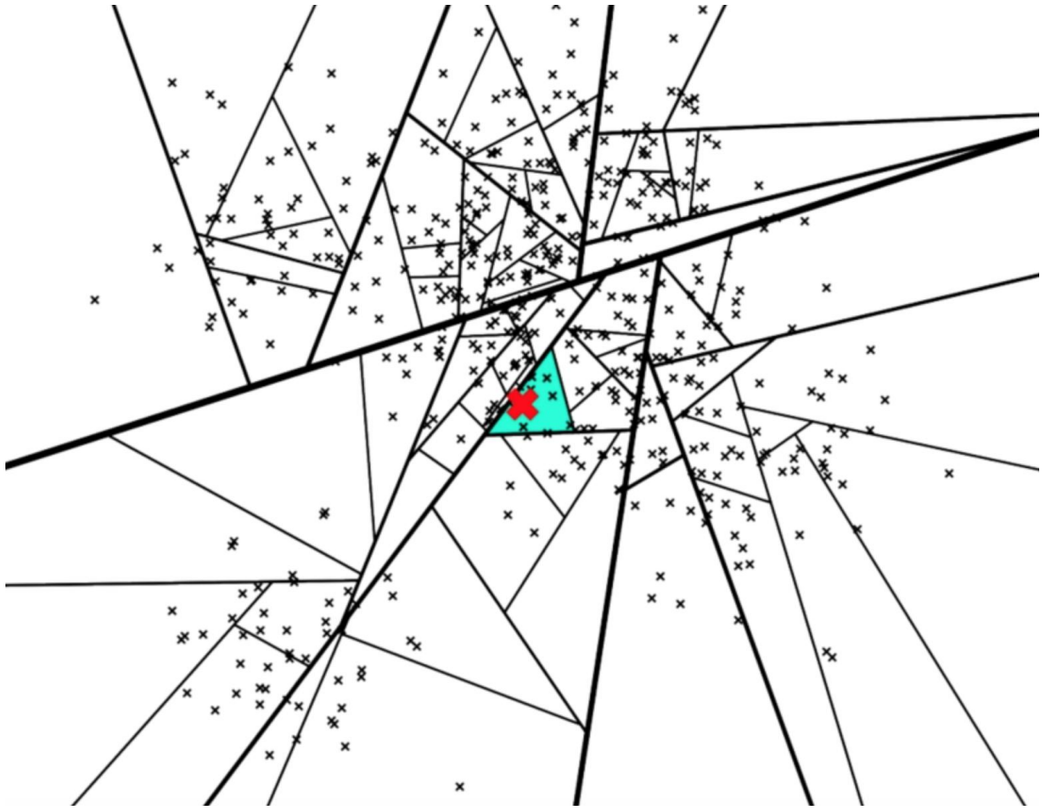




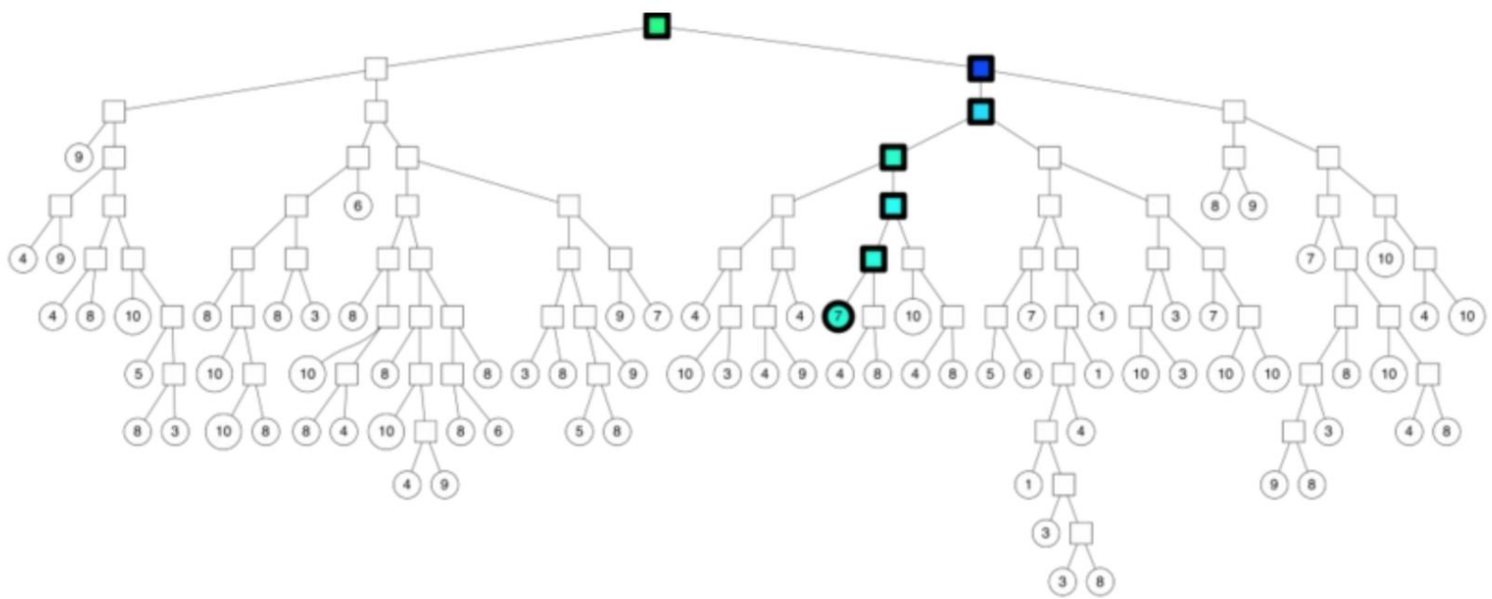
循环直至子节点的样本数小于一定阈值：

1. 在样本空间中随机选择2个点
2. 做垂直平分，将空间一分为二，同时更新二叉树

算法解析



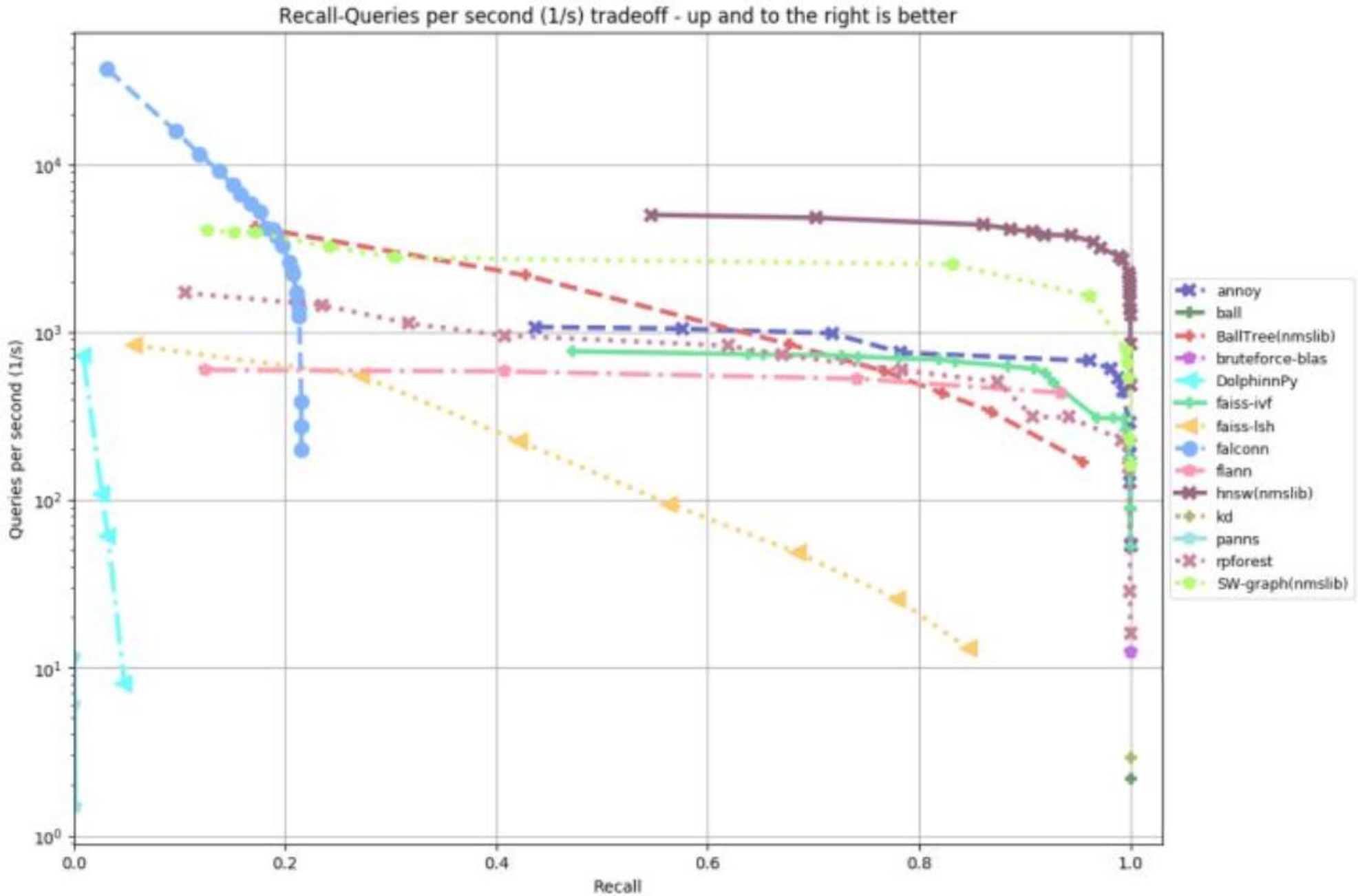
1. 给定一个样本点要查找其最近邻：从根节点出发，一路找根据二叉树找到一个最小划分的子空间，再计算子空间里所有点的距离
2. 邻近的子空间里的点，其实也离得很近，做法是建立多棵树（建树过程有随机性），在多棵树同时搜索（可以并行），最后融合结果



实际效果

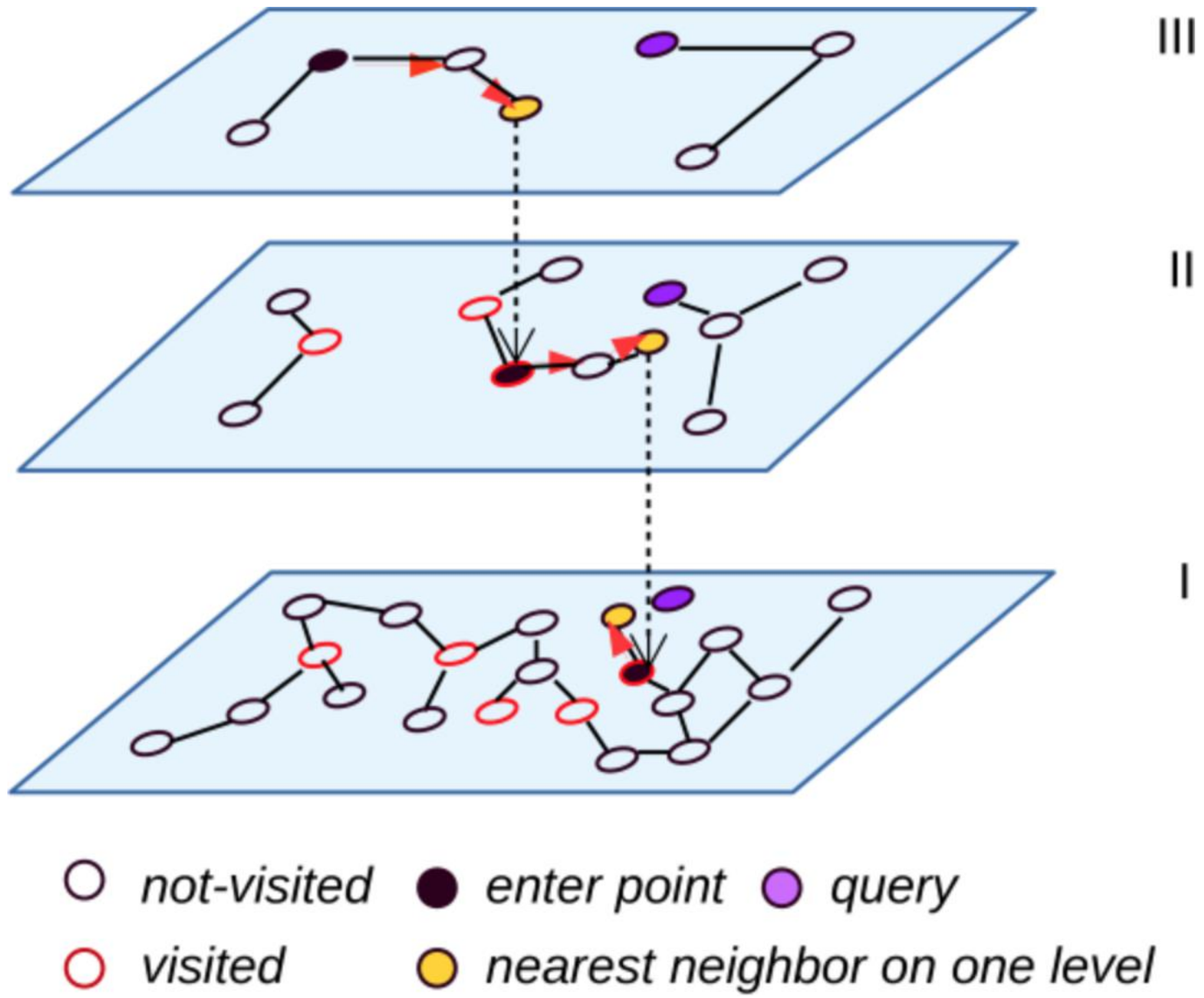
- 召回率较优，和暴力搜索法相比较基本一致
- 建树的速度
  - 千万量级的item，时间为若干小时，尚可以忍受
  - 以100棵树为例，索引文件大约几个G
  - 多核利用支持的不是很好
- 查询的速度很快

HNSW算法

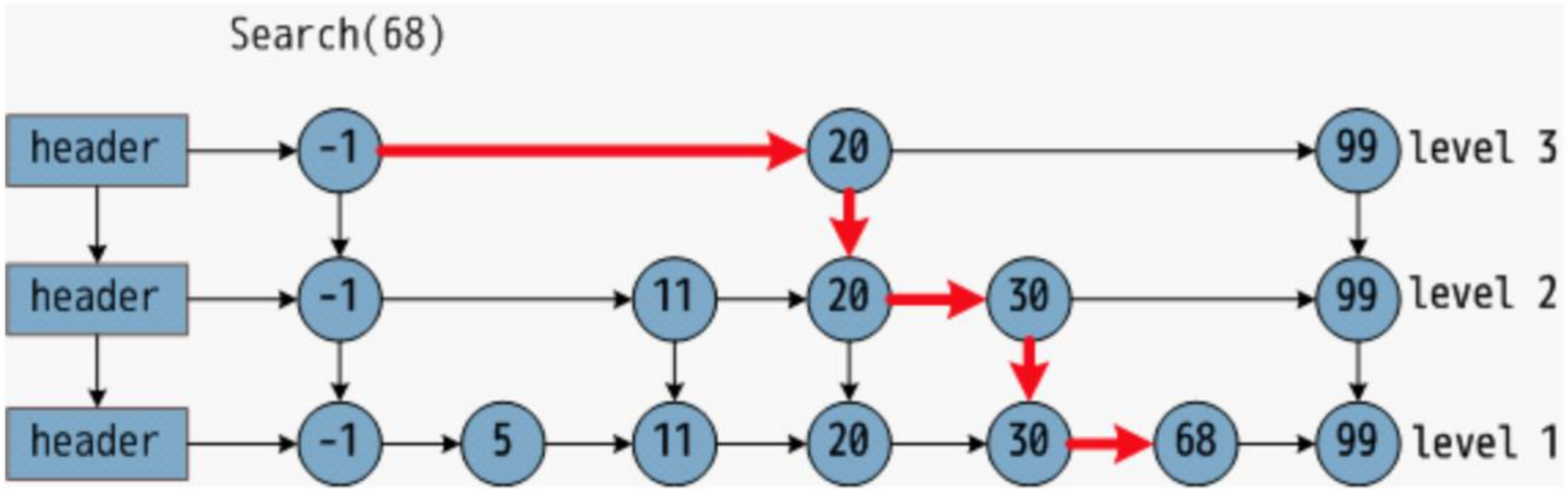


glove-25-angular

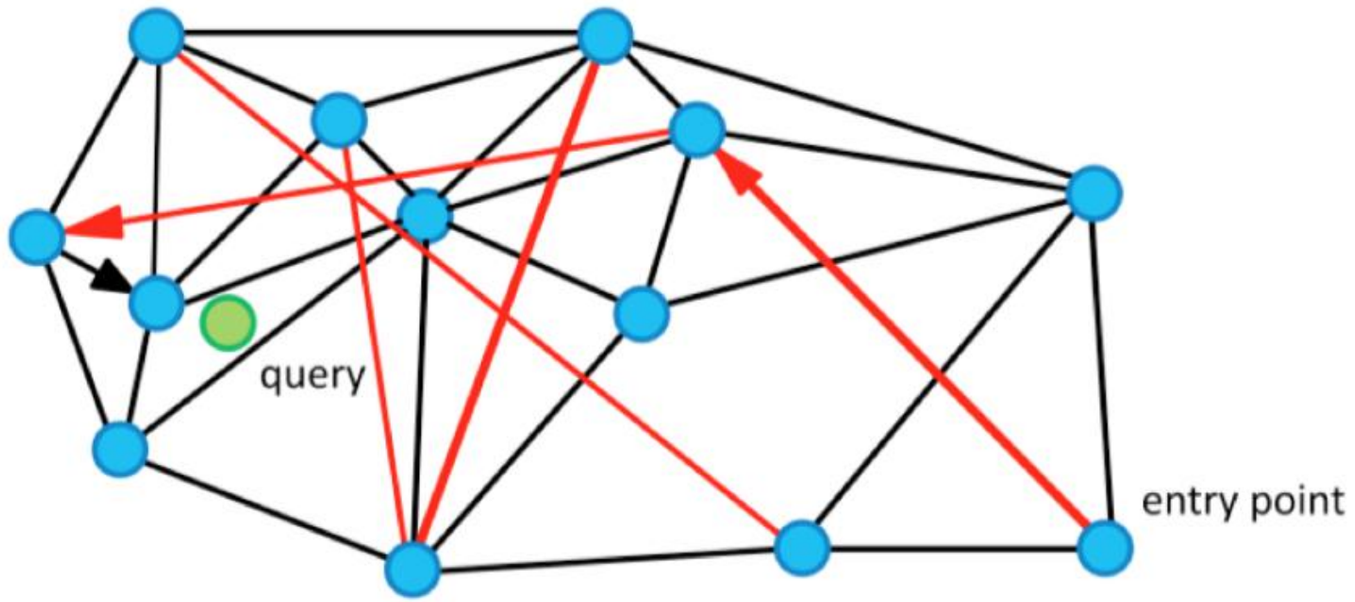
算法解析-检索过程


III
II
I
not-visited
enter point
query
visited
nearest neighbor on one level

算法解析-跳表结构


Search(68)
header
-1
20
99
level 3
header
-1
11
20
30
99
level 2
header
-1
5
11
20
30
68
99
level 1

算法解析-构建NSW



插入一个全新点时，查找到与这个全新点最近的m个点，连接全新点到m个点的连线。

实际效果

- 召回率优秀，和暴力搜索基本一致
- 构图的速度很快
  - 千万量级的item，可在分钟级别完成
  - 多核利用也很优秀，当心把机器跑挂
- 查询速度也很优秀

选择索引算法时的一些指导原则

- 有多少向量需要建索引
- 索引database更新的频率需要如何
- 召回率是否满足需求
- 索引算法所支持的距离度量是否适配
- 同样一种算法，不同库的实现效果也不一样

Faiss官方的指南

---

https://github.com/facebookresearch/faiss/wiki/Guidelines-to-choose-an-index

“有没有更好的方式？”

基于SGNS做召回的局限性

- 无监督方式，不是直接优化目标
- 没法结合side information

深度语义匹配模型DSSM

- 主要贡献
  - 利用点击数据（有监督）
  - 利用深度学习（学习能力强）
  - Word hashing（解决大词表问题）

纸上得来终觉浅，绝知此事要躬行

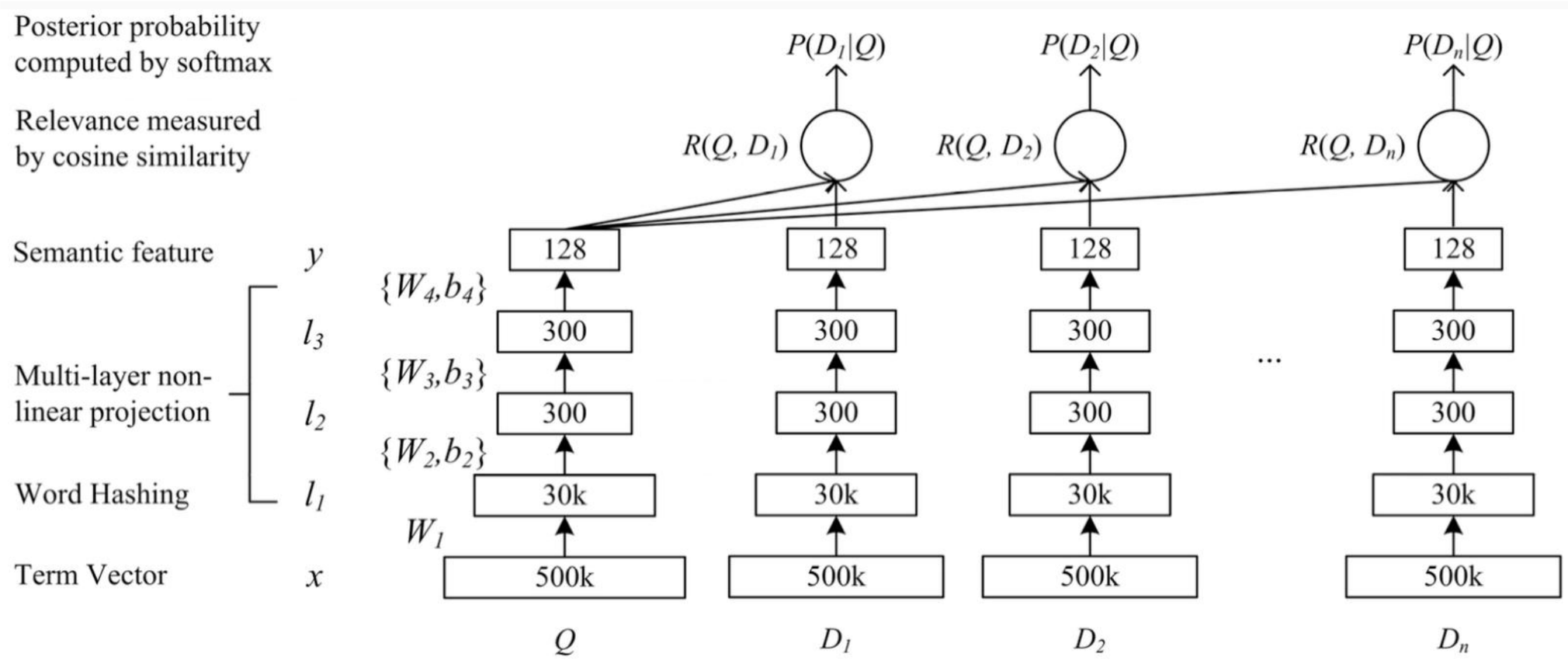
TensorFlow还用介绍吗

TensorFlow 是一个端到端开源机器学习平台

用TF实现DSSM都需要做什么

- 设计和定义模型结构
- 构造训练数据
- 使用GPU进行训练
- 导出训练好的模型
- 导出embedding向量

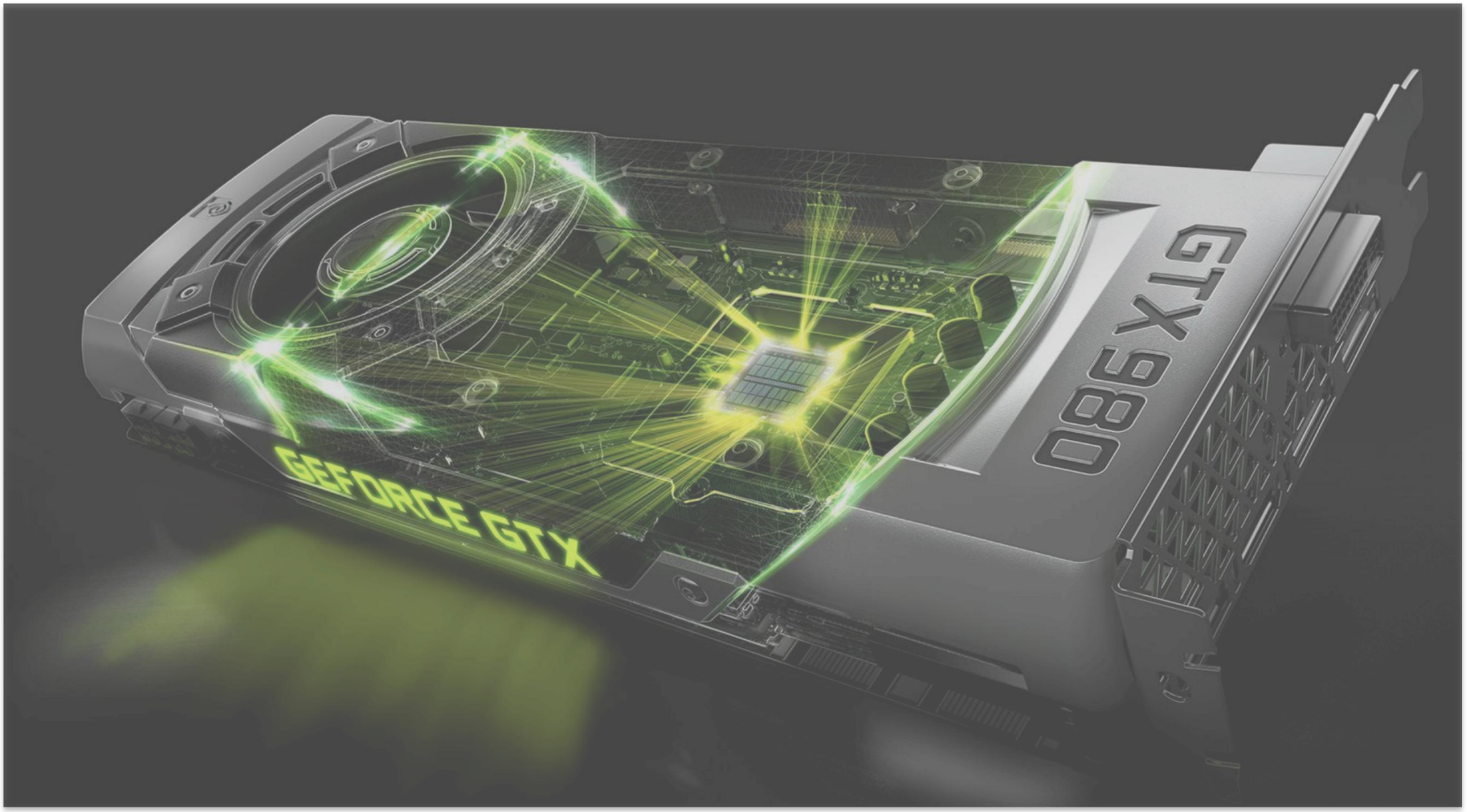
设计和定义网络结构


Posterior probability computed by softmax
Relevance measured by cosine similarity
Semantic feature
Multi-layer non-linear projection
Word Hashing
Term Vector
y
l_3
l_2
l_1
x
$\{W_4,b_4\}$
$\{W_3,b_3\}$
$\{W_2,b_2\}$
W_1
$P(D_1|Q)$
$P(D_2|Q)$
$P(D_n|Q)$
$R(Q, D_1)$
$R(Q, D_2)$
$R(Q, D_n)$
128
300
30k
500k
...
Q
D_1
D_2
D_n

构造训练数据

- 数据下载
- 格式介绍
- 处理成什么样

一些必要的辅助函数（代码）



如何使用GPU（代码）

导出模型和向量（代码）

“关于模型的讨论。”

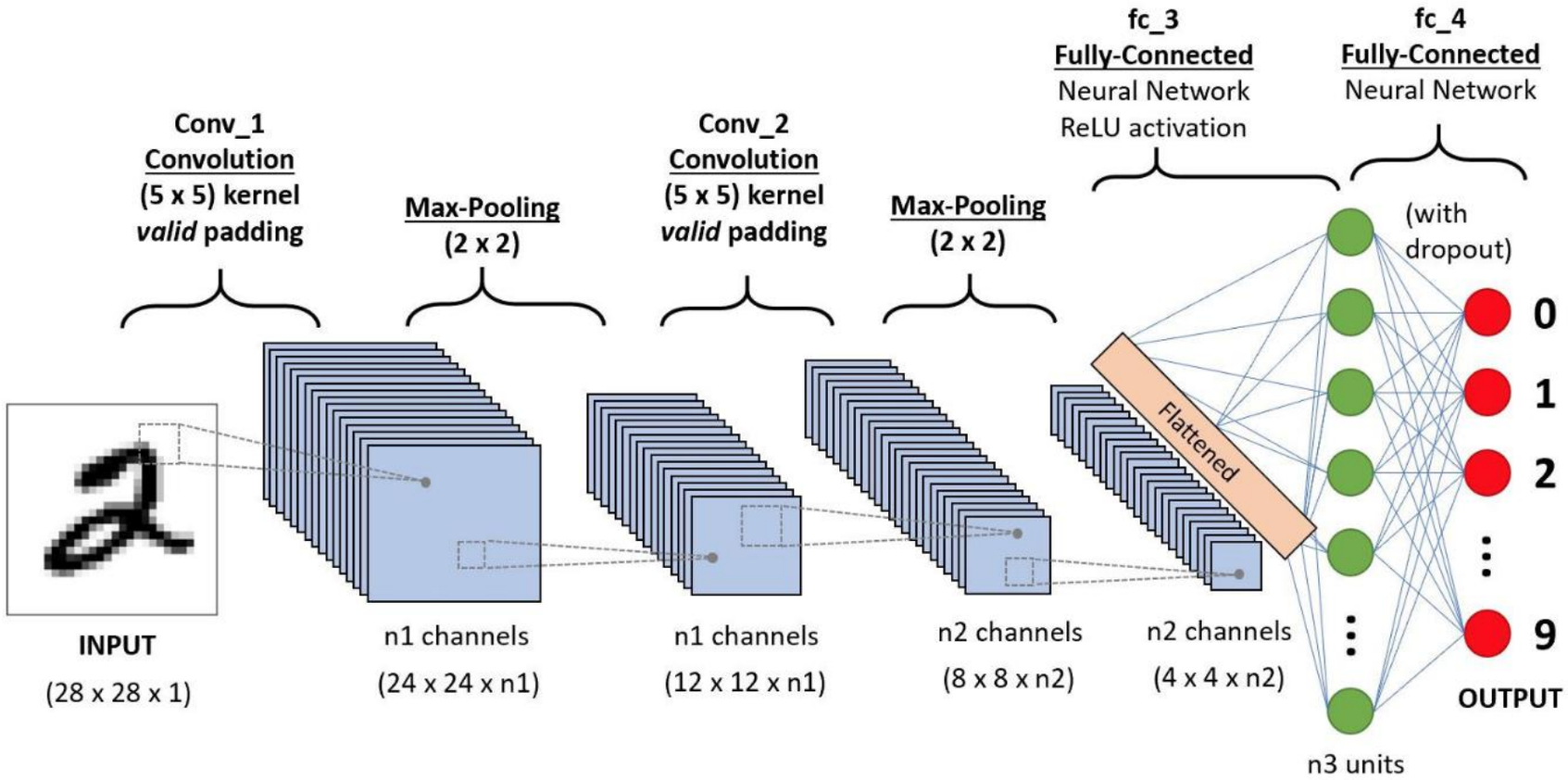
基于DNN的模型有什么缺点

- 同等看待用户的行为，而没有考虑到上下文
- 无法充分反映用户的兴趣

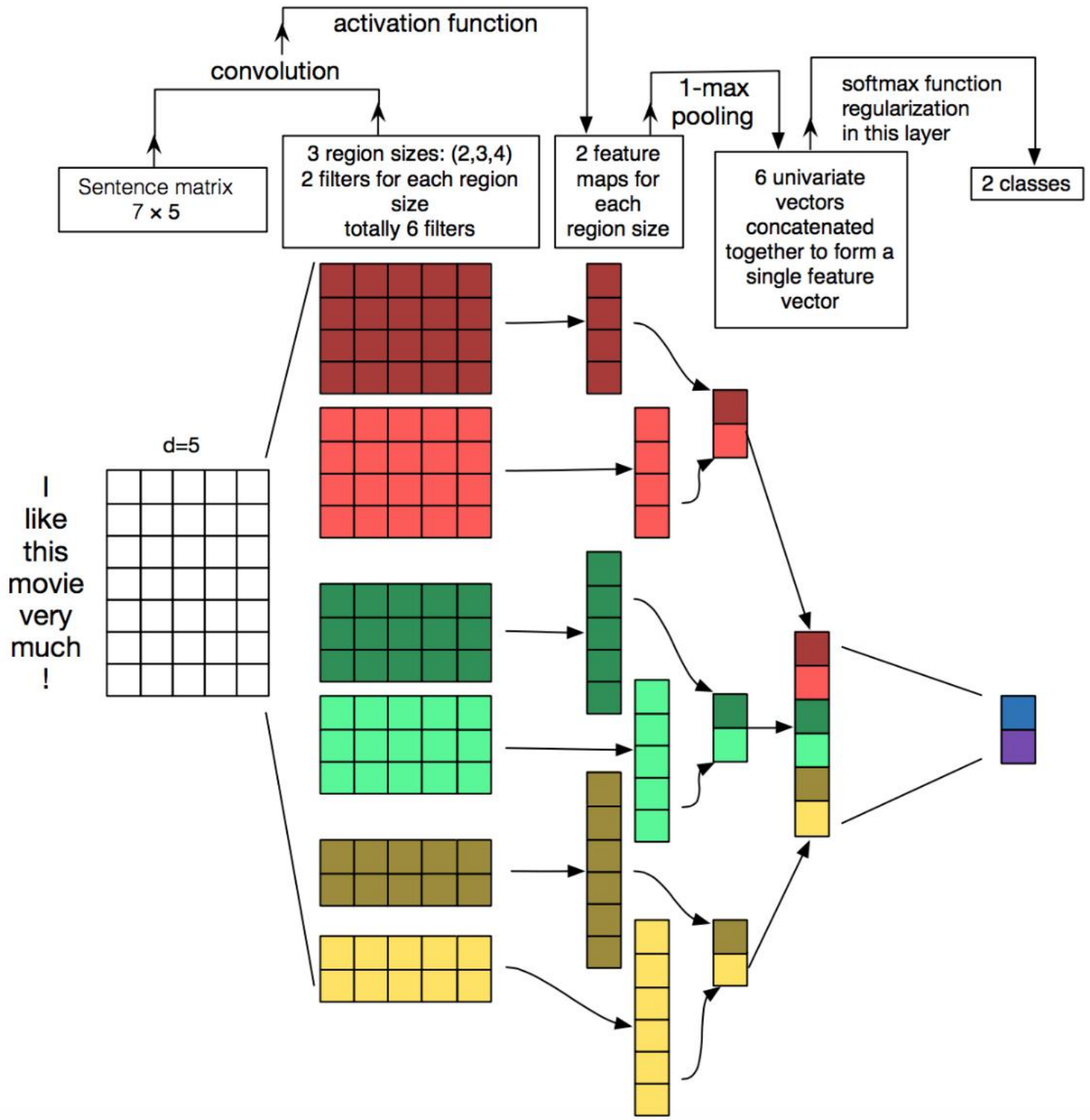
什么叫卷积

$$(f*g)(n)=\sum_{\tau=-\infty}^{\infty}f(\tau)g(n-\tau)$$

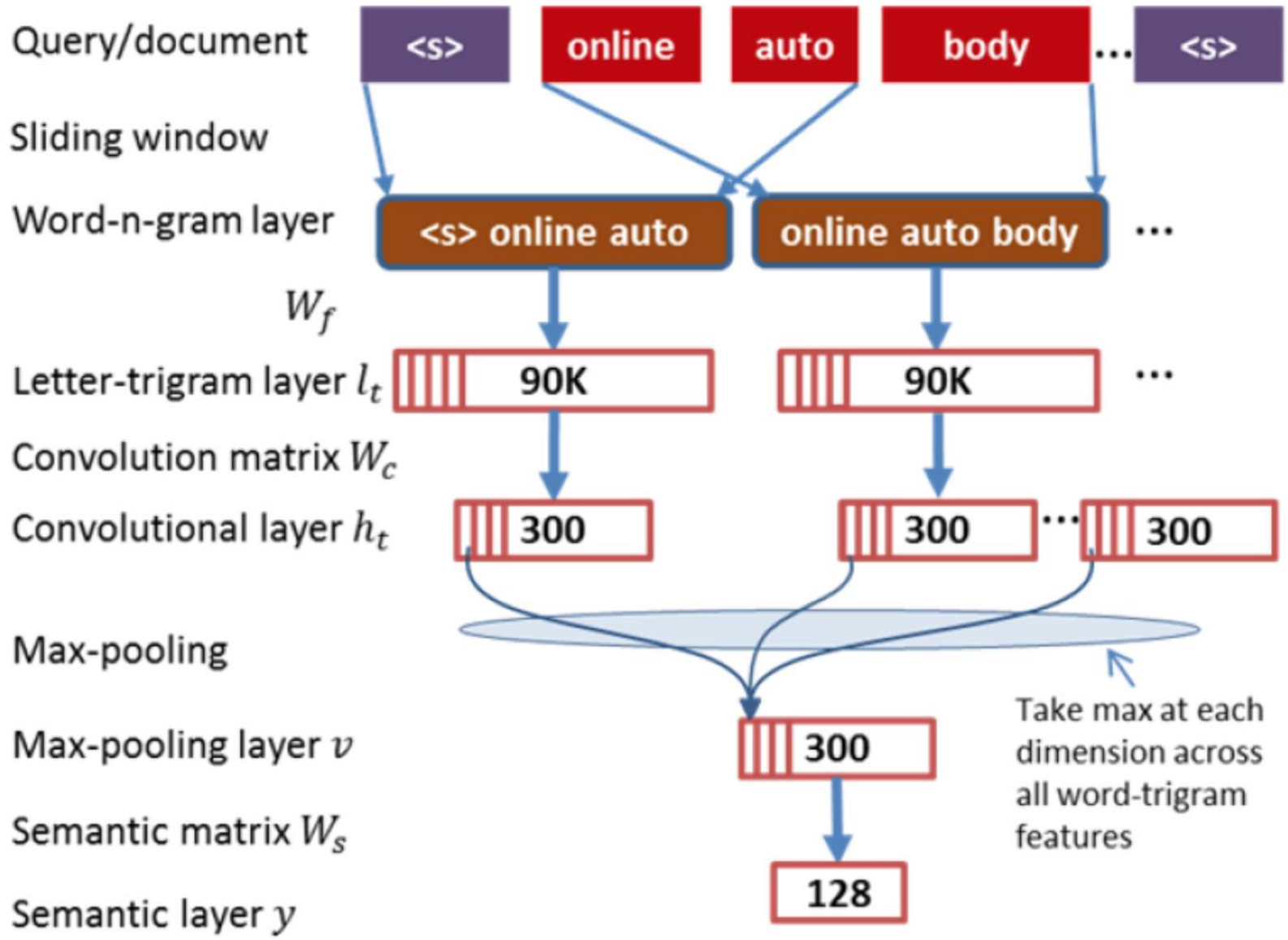
CNN用于图像


Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with
dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

CNN用于NLP


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

CNN改进后的结构


Query/document
<s>
online
auto
body
...
<s>
Sliding window
Word-n-gram layer
<s> online auto
online auto body
...
W_f
Letter-trigram layer l_t
90K
90K
...
Convolution matrix W_c
Convolutional layer h_t
300
300
...
300
Max-pooling
Max-pooling layer v
300
Take max at each dimension across all word-trigram features
Semantic matrix W_s
Semantic layer y
128

“在TF里修改模型结构。”

下一步做什么

- 模型serving
- 确保线上线下数据分布的一致性
- badcase分析
- 调整方案

思考

- 为何深度学习有效
- 如何对行为的时序建模
- 不用深度学习怎么做

总结&回顾

- 承接上回：高维空间向量搜索算法（PQ/Annoy/HNSW）
- SGNS框架的局限性
- 用TF实现一个DSSM模型
- 如何利用CNN结构进行改进
- 理论提升