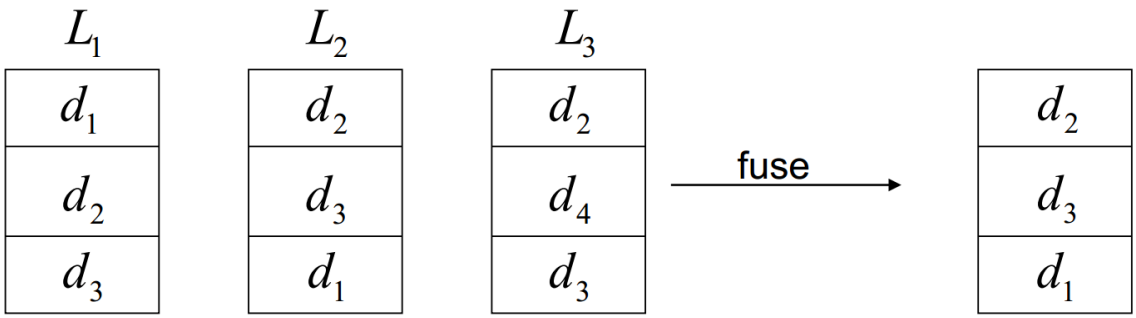
搜索结果融合

信息检索 信息融合 机器学习

搜索结果融合是将不同来源的结果进行合并融合，产出最后的搜索结果，包括: 聚合不同的文档(比如文本，图片，视频等)，不同的Query召回的结果和不同的排序函数得到的排序结果。可以应用到的场景: 搜索结果多样性，专家搜索，评估，查询表现预测，相关性反馈。



如图所示，L_1，L_2，L_3分别代表不同的结果拉链，其中d_1，d_2，...为拉链中的文档，多条拉链的结果经由融合过程后最后生成一条拉链即最终结果。

- 搜索结果融合
  - 理论基础
    - 计算上的社会选择理论
      - 投票机制
  - 融合假设
  - 融合框架
  - 融合分类
    - 不同集合
    - 不同系统
      - 以分值为基础的融合
      - 以排序位置为基础的融合
      - 以文本为基础的融合
    - 不同主题
- 机器学习在融合中的应用
  - 学习方法
    - ProbFuse方法(概率融合方法)
    - SegFuse方法
    - SlideFuse方法

理论基础

计算上的社会选择理论

将社会选择理论应用到计算问题上(比如将投票机制利用在排序聚合/融合上)，再利用计算框架分析和发明社会选择机制。

投票机制

4	3	2	2
Peter	Paul	Paul	James
Paul	James	Peter	Peter
James	Peter	James	Paul

Condorcet原则：Condorcet原则是指存在2个以上候选者时，若存在某个候选者能按过半数规则击败其他所有候选人则称该候选者为Condorcet候选人。

> 由上图可以看出一共有11次投票，其中4次Peter是在第一位的，因此Peter与Paul的对比结果为，在其中四次中Peter排名第一战胜Paul，在最后的两次中Peter排名第二也战胜了Paul，因此，Peter以6次战胜Paul超出了11次投票的半数，所以Peter胜出；Peter与James的对比结果中，其中四次Peter排名第一战胜James，在第三列中Peter又两次战胜James，因此同样以超出半数战胜了James，因此Peter成为Condorcet候选人。

Plurality原则：候选者在所有列表中排名第一的次数，次数最多的即为胜者。

> 可以看到Peter排名第一的次数为4，Paul排名第一的次数为5，James排名第一的次数为2，因此可以得出Paul为Plurality胜者。

Copeland原则：候选人两两比较，利用胜出的次数减去失败的次数，以此值决定胜者。

>

> Peter对Paul胜出6次，失败5次，因此其值为1，Peter为胜者；Peter对James胜出6次，失败5次，因此取值为1，Peter为胜者。

Borda原则：一个候选人在一条拉链的分值是在一条拉链中排在它下面的候选人的个数。

> Peter在四条拉链中得分为3分，3条拉链中得分为1分，在4条拉链中得分为2分，因此其最终得分为23分；Paul在四条来安中得分为2分，在五条拉链中得分为3分，在2条拉链中得分为1分，因此其最终得分为25分；James在四条拉链中得分为1分，在3条拉链中得分为2分，在2条拉链中得分为1分，在2条拉链中得分为3分，因此其最终得分为18分。因此胜出者是Paul。

融合假设

- **检索结果融合后的效果优于只使用单条拉链的结果**
- **当相关文档的重叠要比不相关性文档的重叠要高时融合才有效**

> 这一点是存疑的: 这种假设只有在对结果做类似Ensemble的处理时才是有效的，但是如果是做多样性处理那这个假设就不必须的

- **当拉链的顶部存在独有的相关Doc时融合有效**

融合框架

- **证据推理**
  做法: 用符号表示检索拉链中知识信息，比如拉链中文档的排序位置和它们的分数，在文档的标题和摘要中的Term等
- **几何概率模型**
- **统计原则**
- **概率框架**
- **学习框架**

融合分类

不同集合

不同系统

对不同系统返回的结果，其分值是不可比的，因此，需要将分值进行归一化，做到不同系统返回的结果具有可比性。进行分值归一化的方法包括下面三个:

- Min-Max方法，即 $Norm(Score) = \frac{Score_i - MinScore}{MaxScore - MinScore}$
- Sum Norm方法，即$Norm(Score) = \frac{Score_i - MinScore}{\sum_{j=1}^{n} Score_j - MinScore}$
- Z-Score方法，即$Norm(Score) = \frac{Score_i - u}{\sigma}$，其中u为拉链中文章的分值平均值，$\sigma$为文章分值的标准差

以分值为基础的融合

名称	公式	解释
CombSUM	$\sum_{L_i:d \in L_i} S_{L_i}(d)$	将文档在各个拉链中的分值相加
CombMNZ	$m \cdot \sum_{L_i:d \in L_i} S_{L_i}(d)$	将文档在各个拉链中的分值相加并乘以文档出现在拉链中的次数
CombANZ	$\frac{1}{m} \cdot \sum_{L_i:d \in L_i} S_{L_i}(d)$	将文档在各个拉链中的分值相加并乘以文档在拉链中出现次数的倒数
Linear	$\sum_{L_i:d \in L_i} w_i \cdot S_{L_i}(d)$	将文档在不同拉链中的分值乘上不同的权重

以排序位置为基础的融合

名称	公式	解释
Borda	$\sum_{L_i:d \in L_i} \frac{n - r_{L_i}(d) + 1}{n}$	将每条拉链中排在该Doc以下的Doc数加1除拉链中的Doc数，并将多条拉链结果加和
RRF	$\sum_{L_i:d \in L_i} \frac{1}{v + r_{L_i}(d)}$	取文档在拉链中的排序位置的倒数，并将多条拉链结果加和
ISR	$m \cdot \sum_{L_i:d \in L_i} \frac{1}{r_{L_i}(d)^2}$	取文档在拉链中排序位置的平方的倒数，并将多条拉链结果加和乘上在拉链中出现的次数
logISR	$logm \cdot \sum_{L_i:d \in L_i} \frac{1}{r_{L_i}(d)^2}$	取文档在拉链中排序位置的平方的倒数，并将多条拉链结果加和乘上在拉链中出现次数的log值
RBC	$\sum_{L_i:d \in L_i} (1 - \phi)\phi^{r_{L_i}(d)-1}$	

以文本为基础的融合

- 一个文档包含的Query Term的个数和他们的距离
- 使用基于标题和基于摘要的特征用于以tf为基础的排序
- 使用标题、摘要和URL为基础的特征: 比如QUery字符的n-gram在标题和摘要中的比例，Query Term和标题中的距离以及URL路径的深度等。

不同主题

机器学习在融合中的应用

将融合问题看做机器学习问题，将多条拉链数据的融合看做求解 $F(d;q)$，其中q代表Query，d代表文档，所以将问题转化为当知道Query时计算Doc的得分的问题。形式化表示:

$$F(d;q)=\sum_{L_i:d\in L_i} S_{L_i}(d)w(L_i)$$

也就是计算Doc在各个拉链中的得分，以及各条拉链在Query下的权重，将两者相乘加和得到最终Doc在Query下的分值。

学习方法

ProbFuse方法(概率融合方法)

该方法中Score部分的计算公式如下所示:

$$S_{L_i}(d)=\frac{1}{k}\frac{1}{|Q|}\sum_{q_j\in Q}\frac{R_{k,q_j}}{R_{k,q_j}+NR_{k,q_j}}$$

其做法是将召回的结果列表划分为多个Block，比如对于召回的结果，我们按照十个结果一组进行划分，那么如果一个Query为q_i召回了100条结果，则可以划分为10个block，之后计算每个Block内结果相关的概率，如果所有数据都有标注，则可以将一个Block内结果相关的概率记为:

$$P(d_k|q)=\frac{R_{k,q}}{All_{block}}$$

即使用该Block中相关的结果除以整个Block的全部结果，之所以在PPT中将Doc相关的概率写成

$$P(d_k|q) = \frac{R_{k,q}}{R_{k,q} + NR_{k,q}}$$

因此，如果存在未标注数据则会被忽略掉，通过将训练集合中所有Query在Block K上的加和平均即可得出召回链i在Block k上结果相关的概率的估计，计算公式为:

$$P(d) = \frac{1}{|Q|} \sum_{q_j \in Q} \frac{R_{k,q}}{R_{k,q} + NR_{k,q}}$$

之后再除以Block在结果中的编号即为该Doc在拉链中的得分，即

$$Score_{L_i}(d) = \frac{P(d)}{k}$$

如果是拉链中的第一个位置，则k的值为1，如果是第二个位置则k的值为2，以此类推。最后通过将多条拉链的分值按照权重加和即可得出Doc的最终得分，通过该分值进行排序。

SegFuse方法

SegFuse看上去像是ProbFuse的一个变种，可以先看下其计算公式:

$$S_{L_i}(d) = (1 + normScore_{L_i}(d)) \frac{1}{|Q|} \sum_{q_j \in Q} \frac{R_{k,q_j}}{All_{k,q_j}}$$

可以看到后面的部分和ProbFuse是一致的，只是在除以k变为了$1 + normScore_{L_i}(d)$，这样其实就相当于在计算normScore的基础上加了一个相关性的概率值，这样比起直接计算normScore要更加准确。(不过这里比较奇怪的是既然参考ProbFuse为什么还是使用了Block内的全部文档而不是只用有标注的)

SlideFuse方法

在引出SlideFuse之前先介绍下PosFuse，PosFuse方法与上面的两个方法不同，上面两个方法都是对结果进行分块，然后计算块内的相关性概率，PosFuse则是直接计算各个位置上的相关性概率。SlideFuse则是在PosFuse的基础上进行改进，SlideFuse中Doc的分值是以Doc前a个位置和Doc后b个Doc的PosFuse概率的平均值作为Doc相关的概率。

MAPFuse方法

在MAPFuse方法中，权重值为$w(L_i) = \frac{1}{|Q|}\sum_{q_j \in Q} MAP_{q_j}$，Score值为$S_{L_i}(d) = \frac{1}{r_{L_i}(d)}$

MAP的计算公式为: $MAP = \frac{1}{|Q|}\frac{1}{|Position|}\sum_{q_i \in Q}\sum_{j \in Position} Precision_j$

$Precison_j = \frac{TP_j}{TP_j + FP_j}$，其中$TP_j$为前j个结果中相关的结果,$FP_j$为前j个结果中不相关的结果。

即在一次搜索中计算出各个位置的精确率的平均值，最后计算出整个Query集合上的平均值即为MAP的值。

所以，对不同的拉链MAP的值越大，则其权重越高。