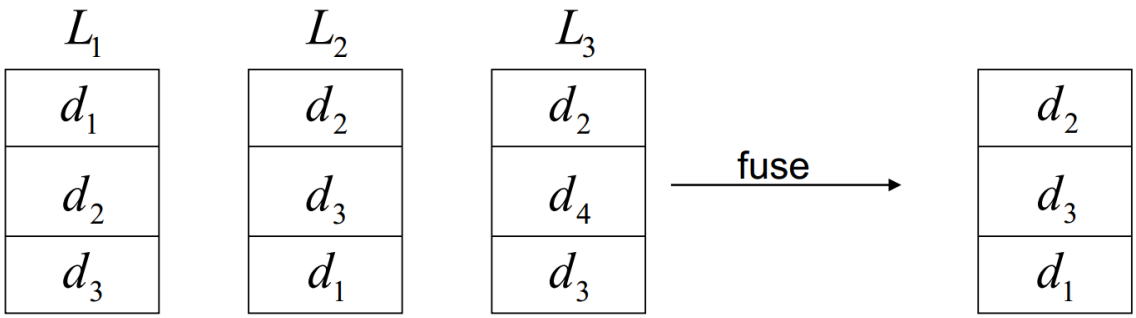
搜索结果融合

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
    - 以分值为基础的融合
    - 以排序位置为基础的融合
      - 位置向分值的转换
    - 以文本为基础的融合
- 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
    - MAPFuse方法
    - 贝叶斯融合方法

LambdaMerge方法
Deep Structured Learning方法

Fusion的应用
多样性
专家搜索
突发时效性融合
评估
Query表现预测
相关性反馈

理论基础

计算上的社会选择理论

社会选择理论是指通过聚合个体表现产出一个集合选择。(比如投票选举)
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。

基于上面的投票方法和社会选择理论给出了几种已有的聚合方法:
1) Condorcet融合方法
2) Kemeny排序聚合方法
Kemeny聚合方法涉及到的一个生成统一排名的算法，称为Kemeny-Young Method。Kemeny-Young Method的思想是找到一个和任何一个排名差距最小的统一排名列表，它通过肯德尔Tau距离作为衡量指标。因为这里使用肯德尔距离是应用于排位，因此不存在序列中存在两个元素值相等的情况(每个Doc肯定排在唯一的一个位置上，不可能存在两个Doc同时排在同一个位置上)，因此在计算肯德尔距离时采用肯德尔相关系数中的公式1即可。计算公式为:

$$Tau - a = \frac{C - D}{\frac{1}{2} N * (N - 1)}$$

计算肯德尔距离的解释: 即生成的统一排序序列中，对于任意的两个文档之间，在统一排序模型中的顺序，在其他被应用的拉链中的的顺序是否与在统一拉链中一致，如果一致则记为一个一致的Pair对，如果不一致则记为一个不一致的Pair对，则比较统一排序序列中顺序的时间复杂度为$O(n^2)$，对于每一对元素在使用的拉链中的比较的时间复杂度则为$O(n)$，且总共存在m条拉链，则每做一次统一排序序列与各个拉链之间的肯德尔相关系数计算的时间复杂度为$O(mn^3)$。【这块没有搞的很明白】

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**

> 该条假设也是为什么我们要进行融合的原因，正是因为我们假设多条拉链融合的结果效果要优于单条拉链，所以，才会促使我们去进行多拉链融合。如果这个假设不成立，那么我们也就没有必要进行融合的工作了。

- **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**

这里总共提出了3个假设，第一个假设是融合值得做的基础，正是因为融合后的效果要好于任何的单一拉链结果，所以我们才有必要做融合，但是何时融合才是有效的，就引出了下面的两个假设，第二个假设中假设只有在不同拉链中相关的内容重叠的数量多于不相关内容数量时才有效，这一点主要是基于投票理论或者Ensemble的方式进行处理时才有效，即当不同拉链中相关内容的重叠多于不相关内容时，此时基于投票理论，相关的内容就更容易被排到前面，对提升最终拉链的相关性就很有帮助，但是这个假设是有问题的，假设存在两条拉链前K条结果是一致的，这样的融合就不会带来任何的改进。这也就引出了下面第三个假设，当拉链顶部存在独有的相关性文档时融合才有效，其实这块可以理解为这种情况下主要是增加了拉链结果的多样性效果。所以，上面的三个假设主要在讨论为什么需要进行融合以及融合在何时有效。

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**
  将每条拉链表示成一个标识拉链中Doc相关性概率的向量，计算一条拉链有效性的方法是计算概率向量与表示概率为TRUE的向量之间的欧几里得距离，也就是说计算表示该拉链的概率向量与一条值全部为1的向量之间的欧几里得距离。
  欧几里得计算公式为:

$$Distance = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- **统计原则**
- **概率框架**
  我们首先列出概率框架的公式，公式看着会比较复杂，但是在理解和计算上并不难。

$$\hat{p}(d|q,r) = \int_{\theta_q} p(\theta_d|\theta_q,r)p(\theta_q|q,r)d\theta_q$$

$$\hat{p}(d|q,r) \approx \sum_{i=1}^{m} p(d|L_i,r)p(L_i|q,r)$$

  其含义就是在q的第r个位置文档相关的打分，该概率的计算方法为计算在q的第r个位置选取第i条拉链的概率及第条拉链的第r个位置相关的概率的乘积，将各条拉链计算出的值加和即可计算出当前q的第r个位置以当前各拉链合并可以得到的相关性得分。
- **学习框架**

融合分类

融合分类根据划分方式的不同可以分成两类:

- 根据融合内容的不同可以分为: 不同集合内容的融合，不同系统内容的融合，不同主题内容的融合。
- 根据在融合中使用的算分因素的不同可以分为: 根据内容分值进行融合，根据内容排序位置进行融合，基于文本的融合。

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	$\sum_{L_i:d \in L_i} (1-\phi)\phi^{r_{L_i}(d)-1}$	根据RBP评估指标产出，根据几何分布降低文档的权重
Markov Chains		

RBP评估指标(Rank-biased precision)该指标假设用户先浏览排在首位的文档然后依次以概率p浏览下一个，以1-p的概率不在浏览该结果列表。则对于长度为L的结果列表，RBP的定义为:

$$RBP = (1-p)\sum_{n=1}^{L} r_n p^{n-1}$$

其中的r_n值表示人工评分，与NDCG中的Rel是相同的。其中的p值是由人工进行指定的。

位置向分值的转换

方式1: 用列表长度进去文档的排序位置即为该文档的得分。

$$|L_i| - r_{L_i}(d)$$

方式2: 用1减去文档位置减去1除以文档列表长度即为文档得分，该分值的取值范围为(0, 1]

$$1 - \frac{r_{L_i}(d) - 1}{|L_i|}$$

方式3: 文档在列表中的排序位置加上一个平滑系数的倒数。

$$\frac{1}{\nu + r_{L_i}(d)}$$

方式4: 1加上列表长度的谐波数减去文档在列表中排序位置的谐波数。

$$1 + H_{|L_i|} - H_{r_{L_i}(d)}$$

谐波数在Wiki上的解释: 在数学上，第n个谐波数是前n个自然数的倒数之和。

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}$$

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
- 使用参考term的统计作为文章统计，Term的权重根据与文章开头的偏置设定
- 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。

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。

贝叶斯融合方法

计算一个Doc是否是相关的，则有如下公式表示一个Doc相关的概率为:$P(Rel|d)$，进一步可以表示为:

$$P(Rel|d) = P(Rel|r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d))$$

即在已知Doc在各条拉链中排序位置的前提下，文档相关的概率是。同时也可以得到文档不相关的概率公式为:

$$P(NRel|d) = P(NRel|r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d))$$

则我们可以得到相关概率与不相关概率的比值，以此作为文档的分值，则表示为

$$O(Rel) = \frac{P(Rel|d)}{P(NRel|d)}$$

将上面的$P(Rel|d)$和$P(NRel|d)$带入公式，可以得到

$$O(Rel) = \frac{P(Rel|r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d))}{P(NRel|r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d))}$$

由贝叶斯公式可以得到下面的变换

$$P(Rel|d) = P(Rel|r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d)) = \frac{P(Rel)P(r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d)|Rel)}{P(r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d))}$$

则$O(Rel)$经过变换可以变为:

$$O(Rel) = \frac{P(Rel)P(r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d)|Rel)}{P(NRel)P(r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d)|NRel)}$$

因为$P(Rel)$可以由数据统计得出，并且$r_{L_1}(d)$，$r_{L_2}(d)$，…，$r_{L_m}(d)$的相关性概率是独立的，所以可以得到$P(r_{L_1}(d), r_{L_2}(d), \ldots, r_{L_m}(d)|Rel)$的值为$P(r_{L_1}(d)|Rel) * P(r_{L_2}(d)|Rel) * \ldots * P(r_{L_m}(d)|Rel)$，对$P(r_{L_i}(d))$可以参考上面ProbFuse或者SlideFuse的做法得出。

LambdaMerge方法

Deep Structured Learning方法

Fusion的应用

多样性

专家搜索

突发时效性融合

对于突发事件，我们在排序结果中往往要将该类结果排到前面，根据S. Liang and M. de Rijke提出的方法，可以将该过程分为3步:

1) 对检索返回的文章根据Fusion中提到的CombSum等方法计算出各个文档的分值
2) 根据文档分布的时间戳和分值检测突发事件
3) 根据以下三部分计算出一个新的融合分值: 1. 文档与Query的相关性分值 2. 文档所处的Block与Query相关的概率(参考上面ProbFuse等方法) 3. 文档属于突发事件的概率(如何计算?)
算分公式为:

$$F(q,d) = (1-u)p(d|q) + u\sum_{b\in B} p(d|b)p(b|q)$$

这里的b有两个含义，在$p(d|b)$中表示的是文档属于突发事件的概率，$p(b|q)$中表示的是在block中与Query相关的概率。

评估

现在的检索系统评估一般都是采用Cranfield方式进行，该方式的做法为: 1) 准备一个Query集合 2) 针对选定的Query人工标注与Query相关的检索数据集合，人工给Doc进行评分 3)以标准的Doc构建检索系统，使用构建的Query集合进行检索，并计算评估指标(比如NDCG)

上面的评价指标的主要问题在于需要标注的数据量过大，导致标注成功很高。

通过Fusion的方法进行评估集的构建，可以节省大量的标注成本，具体的做法为:
通过N个已经存在的检索系统，进行结果召回，并对这个N个系统召回结果的前K条作为相关结果，而在圈出结果之外的结果则认为是不相关的，并且对于圈定出的Pooling可以加上人工干预，由人工对圈出的评测集合进行Review，以此来构建评测集。【这种方法在TREC这类竞赛项目中比较合适，因为前期的比赛已经存在了大量表现良好的系统，因此可以进行评测集的构建，在现实中一般难以奏效】

Query表现预测

在不进行相关性判断的基础上对Query检索表现进行预测。通过计算多条拉链结果融合的结果与单条拉链检索结果的相似度作为预测。

相关性反馈

以Fuse融合得到结果列表，以用户的点击反馈对Fuse结果进行优化。