Federated Search

Federated Search是指根据Query意图，请求不同的不同集合数据并最终将不同集合的结果进行融合。因此，Federated Search又可以成为Collection Fusion，即集合融合。与前面将的搜索融合中的方式是不同的，之前介绍的融合方式都是来自于同一集合的数据，采取不同的排序策略等生成的结果链，称为Data Fusion。

Federated Search的组成部分:
1) 集合表示
2) 集合选取
3) 结果合并

Federated Search与Data Fusion融合的最终目的都是将多条结果链的结果合并，其中存在的主要困难是不同结果链的分值之间不可比，因此，最终都需要计算出可比的分值。不同之处在于Data Fusion返回的多条结果链中存在比较多的Overlap，对于Federated Search，不同的结果链之间则存在很少的Overlap甚至完全不存在Overlap。所以两者在最终的融合策略上会有所不同。

集合选取

基于词典的集合选取

将集合看作巨大的词袋，并且根据与Query词典相似度做排序。在线应用的时候，计算Query与集合描述的词典相似度(集合描述一般为人工产出的对集合的描述，对于暗网数据则可以根据数据检索抽样作为集合的描述)。该类方法主要是用于知道数据集相关信息的情况下。

GIOSS

bGIOSS根据满足Query的文档评估数量进行排序。其中用到集合的数据量和term的出现频率。计算的公式为:

$$score = \frac{\prod_{j=1}^{m} f_{t_j,c}}{|c|^{m-1}}$$

其中$f_{t_j,c}$为在Collection c中第j个Term出现的次数，$|c|$为Collection c中文档的数量。通过计算比值的乘积来估计整个Query取所有Term交集时的概率。

vGIOSS根据Doc与Query之间的Consine相似度之和作为对每个Collection的评估。并以此进行排序。计算公式为:

$$Goodness(q,l,c) = \sum_{d \in Rank(q,l,c)} sim(q,d)$$

其中$sim(q,d)$为计算Query与Doc的Consine相似度，计算的方式可以采取对Query与Title切词后构建向量后计算Consine值，其中的l值是用于做限定的Limitation，如果q和d的相似度低于该阈值则会被忽略。

CORI

CORI算法对每个集合通过利用应用使用改造过的Okapi term频次正则化的贝叶斯网络模型计算出一个分值。计算公式为:

$$T = \frac{df_{t,i}}{df_{t,i} + 50 + 150 * cw_i/avg\_cw}$$

如果Collection内存在的单词数越多，则其分母越大，计算出的T值越小。

$$I = \frac{log(\frac{N_c+0.5}{cf_t})}{log(N_c + 1.0)}$$

如果cf_t，即包含Term t的文档数量越多，则计算出的I值越小。

$$P(t|c_i) = b + (1-b) * T * I$$

$df_{t,i}$是在Collection i中出现term的文档数量，cf_t是包含Term t的集合的数量，N_c是总共可用的集合数量，cw_i是在集合i中总共的单词数量，avg_cw是所有Collection平均拥有的单词数量，b值是为参数值，通常指定为0.4.

从整个公式来看，I有点类似于IDF的意思，即包含Term t的集合数量越少，则I值越大；而T值则有一些TF的含义，如果Collection包含的词数本身较少，但是包含Term t的文档数量较多则会占优势。所以该计算分值就从词在所有集合中的重要性和词在单个文档集合中重要性都有体现。

CVV

利用文档出现在集合中的次数。定义一个文档集合c对一个包含m个term的query的得分为:

$$Coodness(c,q) = \sum_{j=1}^{m} CVV_j * df_{j,c}$$

其中$df_{j,c}$是Query第j个term在Collection中文档出现的频率，CVV_j是第j个Term的提示有效方差，该值能够反映第j个term将集合从其他集合分离开的等级。CV_j的计算公式为:

$$CV_{c_i,j} = \frac{\frac{df_{j,c_i}}{|c_i|}}{\frac{df_{j,c_i}}{|c_i|} + \frac{\sum_{k\neq i}^{N_c} df_{j,c_k}}{\sum_{k\neq i}^{N_c} |c_k|}}$$

从公式可以看出，$\frac{df_{j,c_i}}{|c_i|}$可以看做是对df_{j,c_i}的正则化处理，因为不同文档集中的文档总量是不一致的，因此直接对比某个term在某集合内的文档数，不具有可比性，因此通过将term在某集合内的文档数正则化后将各个文档集可比，计算CV值，可以看出CV值越大则表示某个文档集内该term在文档集出现的文档占比就越高，也就越容易有好结果【不过这里存在一个风险，如果一个文档集内文档量本身比较小，那么就可能导致召回的结果量不足】。
CCV_j的计算公式为:

$$CCV_j = \frac{\sum_{i=1}^{N_c}(CV_{c_i,j} - \bar{CV_j})^2}{N_c}$$

通过CCV_j计算出Query中各个Term的重要意义，如果CCV_j越大则表示该Term就越具有区分性，因此如果某个集合中存在该Term的文档数量比较多就应该给该文档集打一个更大的分。

文档代理方法

该类型的方法主要用于非合作数据集(即对于数据集的信息是未知的)。该方法同样也可以用于合作数据集。该类方法不仅仅基于Query与集合表述的相似度计算排列，还会用到从集合中采样的文档的排序。该类方法在我们的场景中应用较少，因此，暂时略过。

结果合并

在Federated Search中，一般假设不同的集合之间是不存在Overlap或者重合非常小。Federated Search中的主要任务是为不同集合返回的数据计算出一个可比的分值。

CORI合并策略

CORI将集合分与文档得分组合在一起。首先将集合得分做归一化，集合得分为上面集合选取中提到的每个集合的得分，归一化使用的Min-Max方法。

$$Score_{c'} = \frac{Score_c - Score_{min}}{Score_{max} - Score_{min}}$$

最终CORI方法的组合得分为:

$$Score_{d'} = \frac{Score_d + 0.4 * Score_d * Score_{c'}}{1.4}$$

SSL

SSL是一个半监督学习方法，SSL为每个集合训练一个回归模型将文档的得分映射到全局得分，在SSL方法中除了向各个集合发送查询请求外，还建立一个中心引擎，该引擎中的文档集来自于对所有集合中文档的采样。因此，在进行检索召回时就存在两个分值:
1.来自各个集合的算分
2.来自中心引擎召回数据的算分
并且因为中心引擎中的数据是由各个集合通过采样而来，假设从各个集合召回的结果与中心引擎召回的结果之间是存在重叠的，因此可以利用这部分重叠数据，计算出各个集合计算出的分值向中心集合计算出的分值的一个映射，如果不同的集合的召回和算分用的是同一套模型，则直接训练一个回归模型做集合与中心引擎之间的映射即可；如果不同集合的召回和算分策略都是不同的，就需要为每个集合与中心引擎之间建立一个映射。因此，最终的全局分就是对中心引擎得分近似。通过映射学习出映射参数，从而将无重叠的文档得分也做映射，从而映射出一个全局得分。

但是该方法存在几个问题: 1) 首先需要建立一个中心索引，存在维护成本 2) 中心索引要求所有文档的特征信息可以对齐，因此对可以应用的场景有限制(比如视频、图文内容的很多特征是无法对齐的，无法放在一个中心索引一起算分和召回) 3) 最终分值向中心索引做映射，就需要中心索引的分值计算很准确，否则最终映射出的分值就没有意义了。

扩展: 看上去可以通过对大量Query从各个集合进行召回，然后构建出一个训练集，由人工进行标注打分，之后再通过利用选取的Query从中心引擎和各个集合召回，根据Overlap计算出映射分值。这样可以使得中心引擎得到的得分是人工标注的，分值可以比较准确，但是这里会引入标注代价。