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)。

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，$avg_c w$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:

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