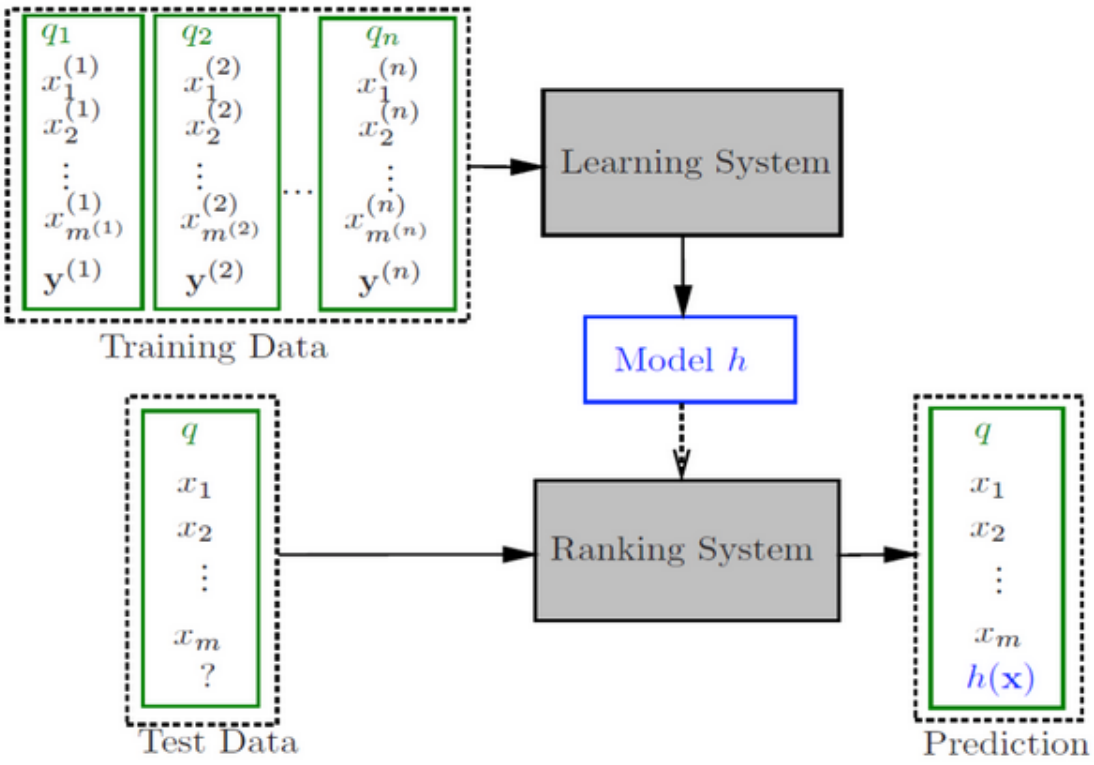
排序学习:Learning to Rank

一.LTR简介

利用机器学习技术来对搜索结果进行排序，这是最近几年非常热门的研究领域。**较典型的是搜索引擎中一条查询query，将返回一个相关的文档document，然后根据(query,document)之间的相关度进行排序，再返回给用户。**而随着影响相关度的因素变多，使用传统排序方法变得困难，人们就想到通过机器学习来解决这一问题，这就导致了LTR的诞生。

1.1 LTR结构



所描述的步骤为：**训练数据获取->特征提取->模型训练->测试数据预测->效果评估。**
接下来，我们依次描述上述的步骤。

二.训练数据的获取

2.1 人工标注

人工标注的数据主要有以下几大类型:

- 单点标注
  - 对于每个查询文档打上绝对标签
  - 二元标注：相关 vs 不相关
  - 五级标注：完美(Perfect),出色(Excellent),好(Good),一般(Fair),差(Bad) ，一般后面两档属于不相关
    - 好处：标注的量少O(n)
    - 坏处：难标。。。不好统一
- 两两标注
  - 对于一个查询`Query`,要标注文档d1比文档d2是否更加相关 (q,d1)>(q,d2)?
    - 好处：标注起来比较方便
    - 坏处：标注量大 估计得有O(n^2)
- 列表标注
  - 对于一个查询`Query`，将人工理想的排序整个儿标好
    - 好处： 相对于上面两种，标的效果会很好
    - 坏处： 标注量极大。

2.2 搜索日志

当搜索引擎搭建起来之后用户的点击数据就变得非常好使。

比如，**结果ABC分别位于123位，B比A位置低，但却得到了更多的点击，那么B的相关性可能好于A.**

点击数据隐式反映了同Query下搜索结果之间相关性的相对好坏。在搜索结果中，高位置的结果被点击的　概率会大于低位置的结果，这叫做"点击偏见"（Click Bias）。但采取以上的方式，就绕过了这个问题。因为我们只记录发生了"点击倒置"的高低位结果，使用这样的"偏好对"作为训练数据。

在实际应用中，除了点击数据，往往还会使用更多的数据。比如通过session日志，挖掘诸如**页面停留时间**等维度，即认为停留时间长打得分越大。在实际场景中，搜索日志往往含有很多噪音。且只有Top Query（被搜索次数较多的Query）才能产生足够数量能说明问题的搜索日志。

2.3 公共数据集

现存一批公开的数据集可以使用

1. LETOR, http://research.microsoft.com/en-us/um/beijing/projects/letor/
2. Microsoft Learning to Rank Dataset, http://research.microsoft.com/en-us/projects/mslr/
3. Yahoo Learning to Rank Challenge, http://webscope.sandbox.yahoo.com/

三.特征提取

搜索引擎会使用一系列特征来决定结果的排序。一个特征称之为一个"feature"。
按照我的理解，feature可以分为3大类：

1. Doc本身的特征：PageRank、内容丰富度、是否是spam、质量值、CTR等 [PageRank是用户随意点一个网页，更容易点到这个网页的数值]
2. Query-Doc的特征：Query-Doc的相关性、Query在文档中出现的次数，查询词的Proximity值（即在文档中多大的窗口内可以出现所有查询词）等。当然，有些Query-Doc的特征不是显式的，而是有Semantic的，即虽然Query在文档中没有出现，但是语义上是有关系的。
3. Query的特征：Query 在所有Query 中的出现次数、比率等

此阶段就是要抽取出所有的特征，供后续训练使用。

四.模型训练

LTR算法主要包括三种类别：单文档方法（Point-Wise Approach），文档对方法（Pair-Wise Approach)和文档列表方法（List-Wise Approach)。

4.1. Point-Wise

单文档方法的处理对象是单独的一篇文档，将文档转换为特征向量后，机器学习系统根据从训练数据中学习到的分类或者回归函数对文档打分，打分结果即是搜索结果。下面我们用一个简单的例子说明这种方法。

下图是人工标注的训练集合，在这个例子中，我们对于每个文档采用了3个特征： 查询与文档的Cosme相似性分值、查询词的Proximity值及页面的PageRank数值，而相关性判断是二元的，即要么相关要么不相关，当然，这里的相关性判断完全可以按照相关程度扩展为多元的，本例为了方便说明做了简化。

文档ID	用户查询	Cosine 相似度	Proximity值	PageRank值	相关性
1	Mysql 使用	0.03	2	5	相关
2	微软产品	0.24	4	7	相关
3	苹果 电脑	0.19	8	2	不相关
4	苹果 电脑	0.43	3	3	相关
5	上市 公司	0.28	7	1	不相关

例子中提供了5个训练实例，每个训练实例分别标出来其对应的查询，3个特征的得分情况及相关性判断。对于机器学习系统来说，根据训练数据，需要如下的线性打分函数：

- **Score(Q, D)=a *CS+b*PM+c*PR+d**
  cs代表Cosine相似度变量，PM代表Proximity值变量，PR代表pageRank， 而a、b、c、d则是变量对应的参数。

如果得分大于设定阈值，则可以认为是相关的， 如果小于设定阈值则可以认为不相关。**通过训练实例，可以获得最优的a、b、c、d参数组合**，当这些参数确定后，机器学习系统就算学习完毕，之后即可利用这个打分函数进行相关性判断。对于某个新的查询Q和文档D，系统首先获得其文档D对应的3个特征值，之后利用学习到的参数组合计算两者得分，当得分大于设定的阈值，即可判断文档是相关文档，否则判断为不相关文档。

point-wise形式：

- 输入：doc的特征向量
- 输出：每个doc的相关性分数
- 损失函数：回归loss，分类loss，有序分类loss[whats this?]

优点:

- 速度快，复杂度低.

缺点:

- 效果一般
- 没有考虑到文档之间的相对关系
- 在排序中，排在最前的几个文档对排序效果的影响非常重要，Pointwise没有考虑这方面的影响

pointwise常用算法：

- Classification
  - Discriminative model for IR (SIGIR 2004)

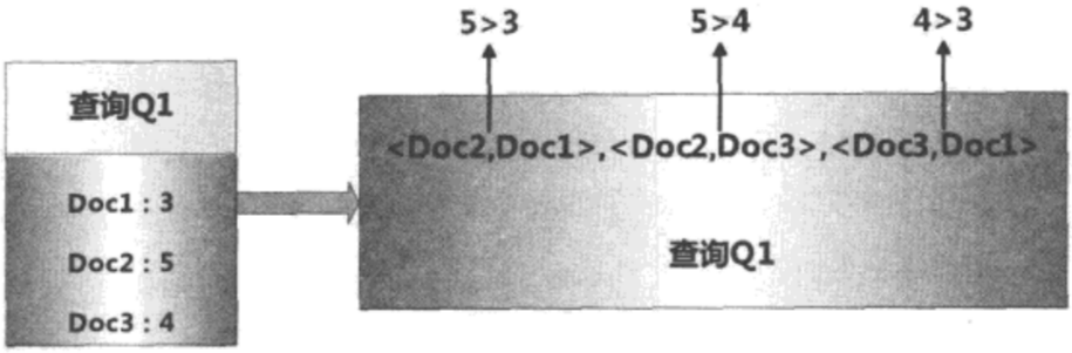
    - **McRank (NIPS 2007)**
- Regression
    - Least Square Retrieval Function (TOIS 1989)
    - Regression Tree for Ordinal Class Prediction (Fundamenta Informaticae, 2000)
    - **Subset Ranking using Regression (COLT 2006)**
- Ordinal Classification
    - **Pranking (NIPS 2002)**
    - OAP-BPM (EMCL 2003)
    - Ranking with Large Margin Principles (NIPS 2002)
    - Constraint Ordinal Regression (ICML 2005)

4.2. Pair-Wise

对于搜索任务来说，系统接收到用户查询后，返回相关文档列表，所以问题的关键是**确定文档之间的先后相对顺序关系**， 而Pairwise则将重点转向对文档关系是否合理的判断。

Pairwise主要是将排序问题转为了文档对顺序的判断。

对于查询`Q1`进行人工标注之后,`Doc2=5`的分数最高，其次是`Doc3`为4分，最差的是`Doc1`为3分，将此转为相对关系之后有:`Doc2>Doc1、Doc2>Doc3、Doc3>Doc1`，而根据这个顺序关系逆向也可以得到相关性的排序顺序，所以排序问题可以很自然的转为任意两个文档关系的判断,而任意两个文档顺序的判断就称为了一个很熟悉的分类问题.



输入:

- 同一查询的一对文档$x_i, x_j, sign(y_i - y_j)$
- 标注两个文档的相对关系，如果文档x_i比x_j更加相关，则$sign(y_i - y_j) = 1$
- 分别保留同一查询下的文档间关系

输出:

- 排序函数给出文档对的计算得分

Pairwise常用算法:

- Learning to Retrieve Information (SCC 1995)
- Learning to Order Things (NIPS 1998)
- **Ranking SVM (ICANN 1999)**
- **RankBoost (JMLR 2003)**
- LDM (SIGIR 2005)
- **RankNet (ICML 2005)**
- Frank (SIGIR 2007)
- MHR(SIGIR 2007)
- GBRank (SIGIR 2007)
- QBRank (NIPS 2007)

- MPRank (ICML 2007)
- **IRSVM (SIGIR 2006)**
- **LambdaRank (NIPS 2006)**
- **LambdaMART (inf.retr 2010)**

虽然Pairwise方法对Pointwise方法做了改进，但是也明显存在两个问题:

1. 只考虑了两个文档的先后顺序，没有考虑文档出现在搜索列表中的位置
2. 不同的查询，其相关文档数量差异很大，转换为文档对之后，有的查询可能有几百对文档，有的可能只有几十个，最终对机器学习的效果评价造成困难。

4.3. List-Wise

与Pointwise和Pairwise不同，Listwise是将一个查询对应的所有搜索结果列表作为一个训练实例，因此也称为文档列方法。

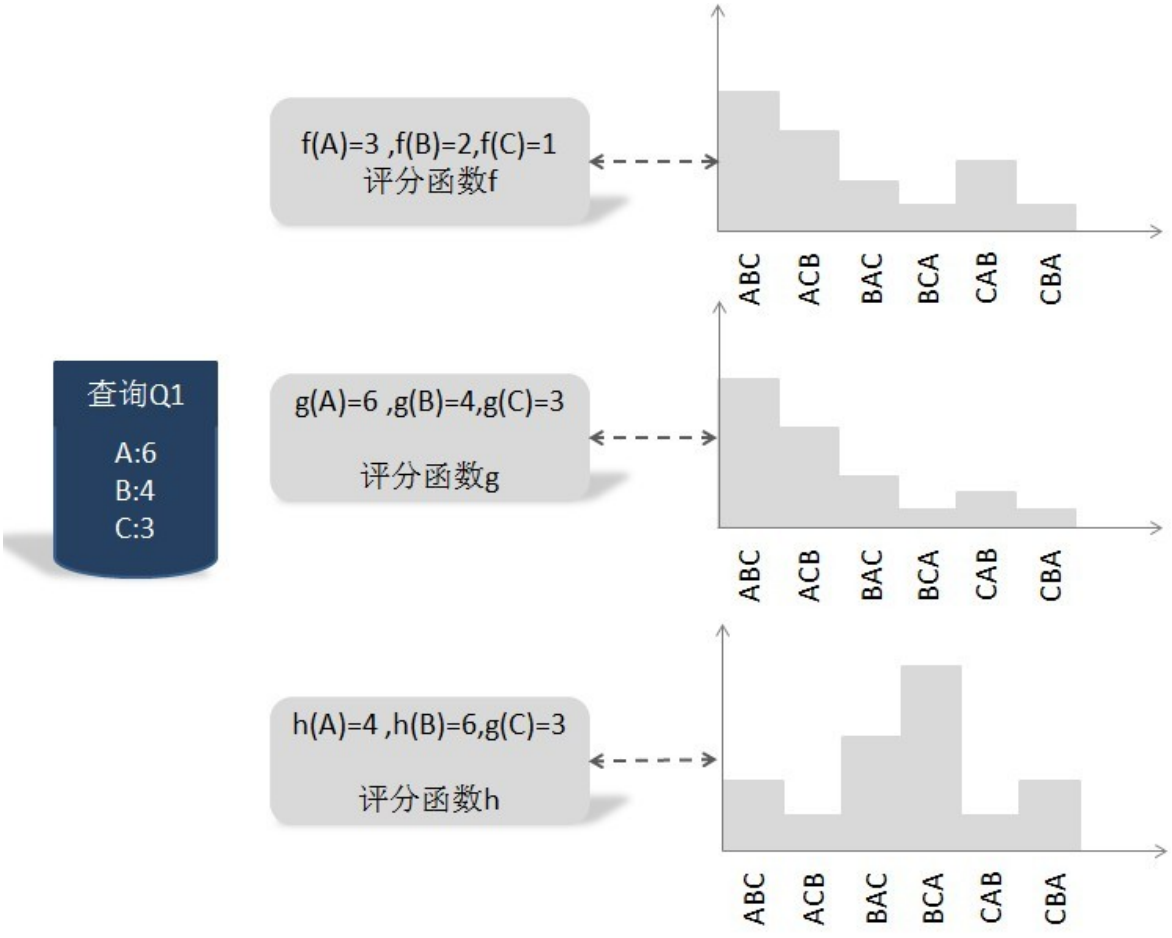
文档列方法根据**K**个训练实例训练得到最优的评分函数**F**，对于一个新的查询，函数**F**对每一个文档进行打分，之后按照得分顺序高低排序，就是对应的搜索结果。所以关键问题是：拿到训练数据，如何才能训练得到最优的打分函数？

Listwise主要有两类:

- Measure specific`: 损失函数与评估指标相关,比如:$L(F(x), y) = exp(-NDCG)$
- Non-measure specific`:损失函数与评估指标不是显示相关,考虑了信息检索中的一些独特性

这里介绍一种训练方法，它是基于搜索结果排列组合的概率分布情况来训练的，下图是这种方式训练过程的图解示意。

这里介绍一种训练方法，它是基于搜索结果排列组合的概率分布情况来训练的，下图是这种方式训练过程的图解示意。



首先解释下什么是搜索结果排列组合的概率分布，我们知道，对于搜索引擎来说，用户输入查询Q， 搜索引擎返回搜索结果，我们假设搜索结果集合包含A. B 和C 3个文档，搜索引擎要对搜索结果排序，而这3个文档的顺序共有6种排列组合方式:

ABC, ACB, BAG, BCA, CAB和CBA,

而每种排列组合都是一种可能的搜索结果排序方法。

对于某个评分函数**F**来说，对3个搜索结果文档的相关性打分，得到3个不同的相关度得分F(A)、 F(B)和F(C)， 根据这3个得分就可以计算6种排列组合情况各自的概率值。 **不同的评分函数，其6种搜索结果排列组合的概率分布是不一样的。**

了解了什么是搜索结果排列组合的概率分布，我们介绍如何根据训练实例找到最优的评分函数。上图展示了一个具体的训练实例，即查询Q1及其对应的3个文档的得分情况，这个得分是由人工打上去的，所以可以看做是标准答案。可以设想存在一个最优的评分函数**g**，对查询Q1来说，其打分结果是：A文档得6分，B文档得4分，C文档得3分， 因为得分是人工打的，所以具体这个函数g是怎样的我们不清楚，我们的任务就是找到一 个函数，使得函数对Q1的搜索结果打分顺序和人工打分顺序尽可能相同。既然人工打分 (虚拟的函数g) 已知，那么我们可以计算函数g对应的搜索结果排列组合概率分布，其具体分布情况如上图中间的概率分布所示。假设存在两个其他函数**h**和**f**，它们的计算方法已知，对应的对3个搜索结果的打分在图上可以看到，由打分结果也可以推出每个函数对应的搜索结果排列组合概率分布，那么**h**与**f**哪个与虚拟的最优评分函数**g**更接近呢？一般可以用两个分布概率之间的距离远近来度量相似性，KL距离就是一种衡量概率分布差异大小的计算工具，通过分别计算**h**与**g**的差异大小及**f**与**g**的差异大小，可以看出**f**比**h**更接近的最优函数**g**，那么在这个函数中，我们应该优先选**f**作为将来搜索可用的评分函数，训练过程就是在可能的函数中寻找最接近虚拟最优函数**g**的那个函数作为训练结果，将来作为在搜索时的评分函数。

上述例子只是描述了对于单个训练实例如何通过训练找到最优函数，事实上我们有**K**个训练实例，虽然如此，其训练过程与上述说明是类似的，可以认为存在一个虚拟的最优评分函数**g** (实际上是人工打分），训练过程就是在所有训练实例基础上，探寻所有可能的候选函数，从中选择那个KL距离最接近于函数**g**的，以此作为实际使用的评分函数。 经验结果表明，基于文档列表方法的机器学习排序效果要好于前述两种方法。

Listwise常用算法:

- Measure-specific
  - **AdaRank (SIGIR 2007)**
  - **SVM-MAP (SIGIR 2007)**
  - **SoftRank (LR4IR 2007)**
  - RankGP (LR4IR 2007)
  - LambdaMART (inf.retr 2010)（也可以做Listwise）
- Non-measure specific
  - **ListNet (ICML 2007)**
  - **ListMLE (ICML 2008)**
  - BoltzRank (ICML 2009)