Performance Evaluation
信息检索绩效评估

• How well does a system perform?	系统的性能如何？
• Two types: quantitative and qualitative	两种类型：定量和定性
• Quantitative: Make numerical measurements	定量：进行数值测量
• Qualitative: Make a judgement	定性：做出判断
• Historically, a heavy emphasis on quantitative measures	从历史上看，重点强调量化措施

Approach to Evaluation
信息检索中的评价方法

Usually the following are used to evaluate:

- A document collection
- A set of test queries
- A set of correct answers to those queries
- A method of scoring

通常以下用于评估：

文件集

一组测试查询

这些查询的一组正确答案

一种评分方法

Procedure of Evaluation
信息检索评估程序

Index the document collection	索引文档集合
Enter each query in turn into the retrieval system	依次将每个查询输入检索系统
Compare the results returned with the ‘correct’ ones	将返回的结果与“正确”的结果进行比较
Score the query according to the prescribed method	根据规定的方法对查询进行评分
Repeat with the other queries	重复其他查询
Combine the scores in some way	以某种方式结合得分

Cyril Cleverdon (1914-1997)
信息检索的先驱

Worked at Cranfield College of Aeronautics

Organised the Cranfield Experiment - the very first Information Retrieval evaluation. 他组织了第一次信息检索评估。

Invented Precision and Recall! 他发明了Precision和Recall！

Cleverdon’s Six Performance Measures
Cleverdon的六项绩效指标

Coverage of the collection – what proportion of all the relevant material it includes	集合的覆盖范围 - 它包含的所有相关材料的比例
Time lag – how long it takes to get an answer	时滞 - 获得答案需要多长时间
Form of presentation of the output	输出的表示形式
Effort involved in obtaining the required answers	获得所需答案有多难?
Recall of the system	“Recall”该系统
Precision of the system	系统的“Precision”

Precision and Recall Evaluation Measures
“P”和 “R”评估措施

Precision	=	*No. Relevant docs returned* / *No. docs returned*	回答查询的系统返回的文档数 / 返回的文件数量
Recall	=	*No. relevant docs returned* / *Total No. relevant docs*	回答查询的系统返回的文档数 / 整个文档集中的文档总数，它们是查询的答案

Precision indicates what proportion of the documents returned are relevant	P表示返回的文件的比例是相关的
Recall indicates what proportion of all relevant documents are recalled	R表示召回所有相关文件的比例
Both precision and recall vary between zero and one	P和R都在0和1之间变化

Precision, Recall and F-Measure
三项评估措施，P，R和F

It is possible to combine Precision and Recall into one measure of a system’s performance.

可以将Precision和Recall结合到一个系统性能的度量中。

The most common way of doing this is van Rijsbergen’s F-Measure which is the harmonic mean of Precision and Recall:

最常见的做法是van Rijsbergen的F-Measure，它是Precision和Recall的调和平均值：

$$F = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Keith van Rijsbergen (mentioned earlier)
F测量的发明者

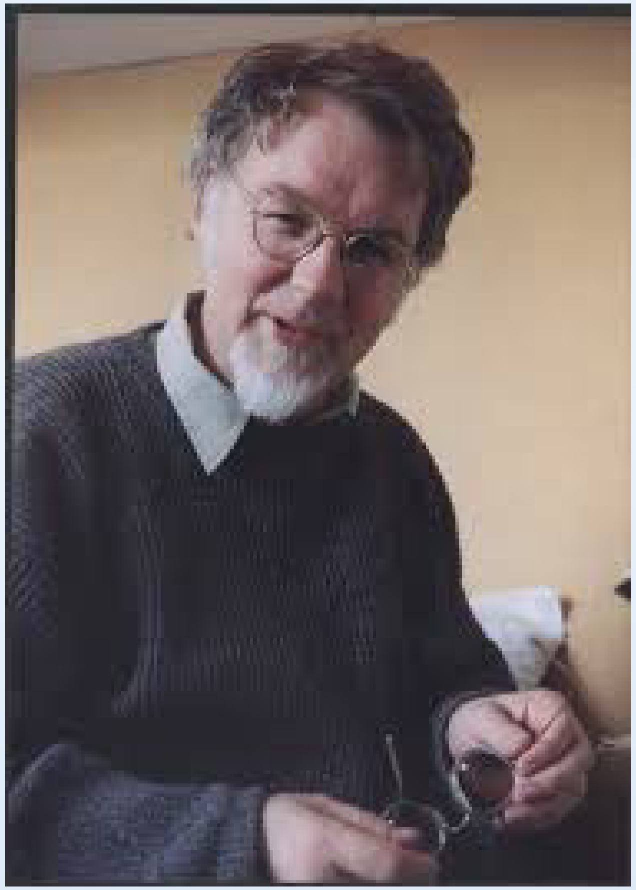
Pioneer of Information Retrieval

Wrote a famous book on IR　他写了一本关于IR的着名书。

Invented Error (E) which combined
P and R
他将P和R结合起来制作F Measure

1-E became F measure!

Book:
http://www.dcs.gla.ac.uk/Keith/Preface.html



F_2 and $F_{0.5}$
F测量中对P和R的不同强调

Sometimes greater emphasis is given to Precision or Recall:

有时更重视Precision或Recall：

$$F_x = (1+x^2) * \frac{Precision * Recall}{(x^2 * Precision) + Recall}$$

Typical values of *x* are 2 and 0.5.

x的典型值是2和0.5。

F_2 will increase the influence of Recall on the combination of Precision and Recall.

F_2会增加Recall的影响力。

This is because as Precision increases, that increase is multiplied by 2^2 and as this is on the denominator, the overall value becomes smaller.

这是因为当P增加时，该增加乘以2的平方，并且因为这是分母，所以整体值变小。

Conversely, $F_{0.5}$ will increase the influence of Precision.

相反，F0.5会增加Precision的影响。

Examples of F Measure
具有不同值的示例

Suppose P=0.8 and R=0.6

F_1 = 2 * 0.8 * 0.6 / (0.8 + 0.6) = 0.69

F_2 = (1+4 * 0.8 * 0.6 / ((4 * 0.8) + 0.6) = 0.63

$F_{0.5}$ = (1+0.25) * 0.8 * 0.6 / ((0.25*0.8) + 0.6) = 0.75

F_1 is weighting P and R equally.

F_2 increases influence of R which is the smaller value at 0.6. Thus F_2 is lower then F_1.

$F_{0.5}$ increases the influence of P which is the larger value at 0.8. Thus $F_{0.5}$ is larger than F_1.

F_1同等地加权P和R。

F_2增加R的影响，这是0.6的较小值。因此F_2低于F_1。

$F_{0.5}$增加了P的影响，这是0.8的较大值。因此$F_{0.5}$大于F_1。

Relationship between Precision and Recall
Precision和Recall之间的联系

A system normally only returns the first *n* documents which match the query	系统通常只返回与查询匹配的前n个文档
If the number of relevant documents is greater than *n,* R cannot be one, even if the system works perfectly	如果相关文档的数量大于n，则R不能为1，即使系统工作正常
Maximum R can be *increased* by making *n* larger	通过使n更大，可以增加最大R.
Maximum R can be *decreased* by making *n* smaller	通过使n更小，可以减小最大R.
This is because with a larger *n,* more relevant documents are likely to be returned	这是因为n越大，可能会返回更多相关文档

Relationship between Precision and Recall cont. Precision和Recall之间的联系

In the extreme case, if all documents in collection are returned for every query, R will always be 1!

在极端情况下，如果每个查询都返回集合中的所有文档，则R将始终为1！

However, an inverse relationship between R and P exists:

但是，R和P之间存在反比关系：

- As R increases, P decreases.

随着R增加，P减少。

When every document is returned, most documents will not be relevant. Thus P will be low.

返回每个文档时，大多数文档都不相关。因此P将很低。

Note on Recall
关于Recall的评论

We can find P by looking at results returned.

我们可以通过查看返回的结果找到P。

However, R depends on knowing every right answer to a query!

但是，R依赖于知道查询的每个正确答案！

We cannot know this unless our collection is small.

除非我们的收藏很小，否则我们无法知道这一点。

We can estimate it by looking at a lot of results returned (e.g. 200).

我们可以通过查看返回的大量结果（例如200）来估计它。

The Pooling method of TREC does this - see later.

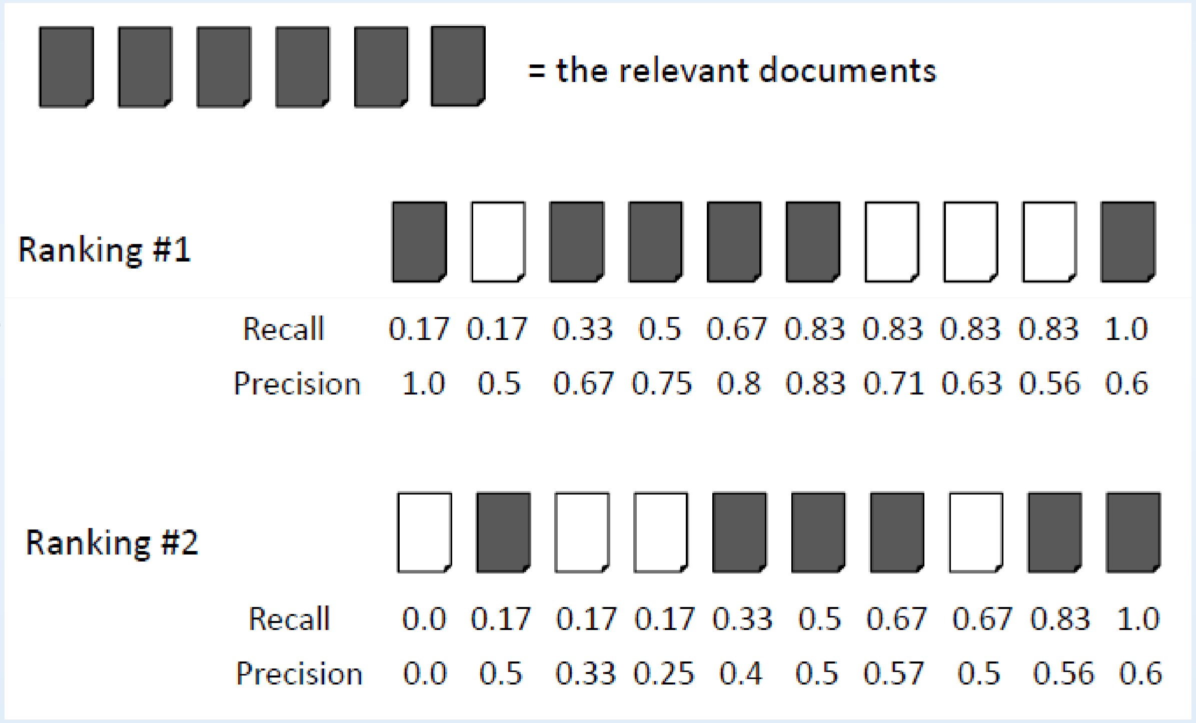
TREC的Pooling方法就是这样 - 见后面。

Which is More Important, Precision or Recall?
哪个更重要，P还是R？

Most searches are on the Web which is very large.	大多数搜索都在网上，这是非常大的。
Returning every document is not possible.	无法返回每个文档。
Instead, answering the user’s question is important.	相反，回答用户的问题很重要。
So P is more important than R.	所以P比R更重要
F gives a good overall idea of performance.	F给出了一个很好的整体性能概念。
Also, F can rank-order the performance of systems e.g. at CLEF or MediaEval.	此外，F可以比较不同参与者系统的性能，例如在CLEF或MediaEval。

Precision and Recall at Different Ranks - 2 Systems, 1 Query
P和R在不同等级测量

Observe how P/R change as more documents returned



■ ■ ■ ■ ■ ■ = the relevant documents

Ranking #1	■	□	■	■	■	■	□	□	□	■
Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2	□	■	□	□	■	■	■	□	■	■
Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

P和R正在发生变化

Comment on Figures
对数字的评论

As docs returned, R increases

当文件被退回时，R增加

P goes up and down as docs returned

当文件返回时，P上下移动

By the end, P/R are the same for both Ranking 1 and Ranking 2

到最后，排名1和排名2的P / R相同

However, Ranking 1 returns the correct documents sooner

但是，排名1会更快地返回正确的文档

How to capture this information?

如何在评分指标中使用此信息？

Precision at Rank
P在给定的等级

Calculate P at standard ranks, e.g. 5 and 10.

Previous example:

System 1:	P@5=0.8	P@10=0.6
System 2:	P@5=0.4	P@10=0.6

Average Precision
平均值P.

Calculate Precision only at ranks where relevant doc returned 仅在返回相关文档的等级处计算精度

Then take average 然后取平均值

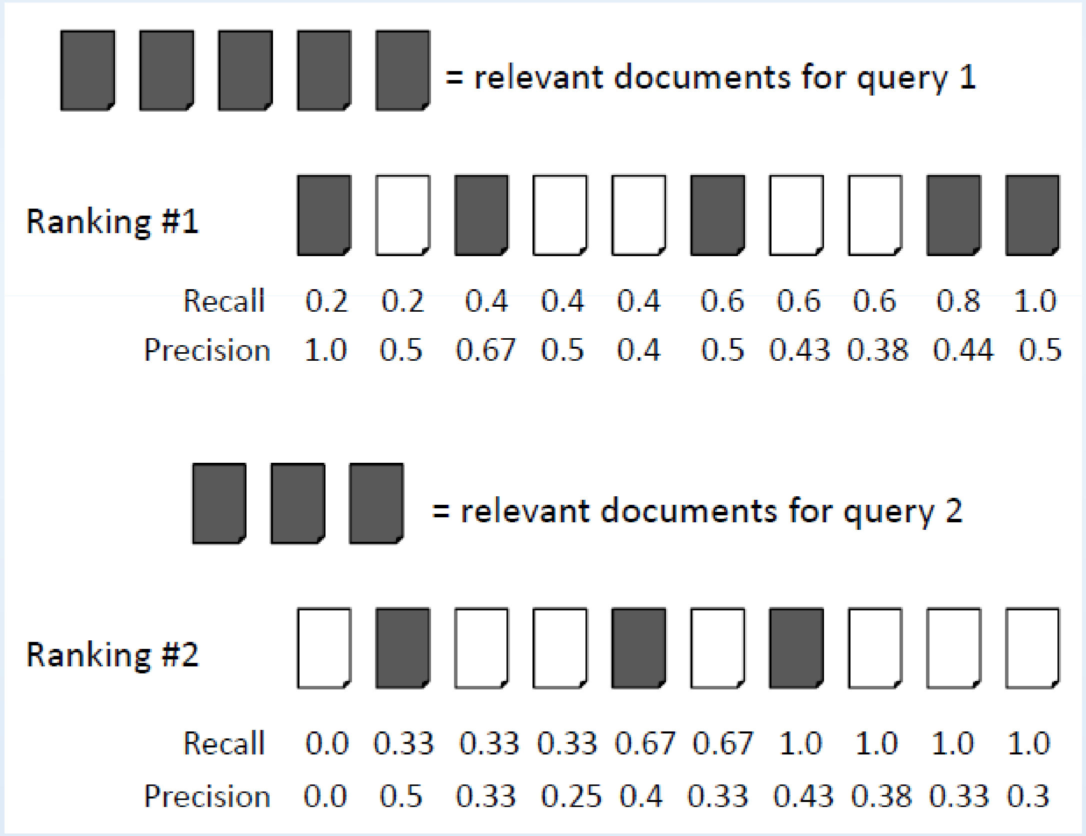
System 1: (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6) / 6 = 0.78
System 2: (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6) / 6 = 0.52

Now we see: System 1 is better than System 2. 现在我们看到：系统1优于系统2

Mean Average Precision Example - 2 Queries, 1 System
平均准确度的平均值 - 注意：2个查询，1个系统

How to combine average precision results for multiple queries?

NB:
2 different queries shown!



■ ■ ■ ■ ■ = relevant documents for query 1

Ranking #1	■	□	■	□	□	■	□	□	■	■
Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5

■ ■ ■ = relevant documents for query 2

Ranking #2	□	■	□	□	■	□	■	□	□	□
Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

如何结合多个查询的平均精度结果？

Calculate Mean Average Precision (MAP)
计算平均P值的平均值

Query 1 average precision: (1.0 + 0.67 + 0.5 + 0.44 + 0.5) / 5 = 0.62
Query 2 average precision: (0.5 + 0.4 + 0.43) / 3 = 0.44
Mean Average Precision: (0.62 + 0.44) / 2 = 0.53

Calculate this value over many queries.

在许多查询中计算此值。

Get one MAP value for system for entire test query set.

为整个测试查询集获取系统的一个MAP值。

This is a very widely used measure.

这是一种非常广泛使用的措施。

Reciprocal Rank
“互惠排名” - 另一项评估措施

Assume you want just one correct answer.

假设您只需要一个正确的答案。

What is the rank of the first correct answer?

第一个正确答案的等级是多少？

Take the reciprocal of that.

取这个等级的倒数

Query 1: dn dr dn dr dn (rank is 2, reciprocal is 0.5)

Query 2: dn dn dn dn dr (rank is 5, reciprocal is 0.2)

For Query 1, first correct result returned sooner.

So, performance for Query 1 is better than for Query 2.

对于查询1，返回第一个正确的结果，排名较低
因此，查询1的性能优于查询2。

Mean Reciprocal Rank
平均互惠排名

Compute Reciprocal Rank (RR) for each test query

Then, compute mean

Called Mean Reciprocal Rank (MRR)

Previous example:

Query 1:	(RR is 0.5)
Query 2:	(RR is 0.2)
MRR:	(0.5 + 0.2) / 2 = 0.35

计算每个测试查询的倒数排名（RR）

然后，计算平均值

称为均值倒数等级（MRR）

Building Test Collections
构建测试集合

Building a Test Collection 1
构建测试集合

Collect documents	收集文件
Collect Queries	收集查询
Experts decide for each query which documents are relevant	专家决定每个查询哪些文件是相关的
Experts may give each relevant document a score	专家可以给每个相关文件一个分数
Comments	
The most thorough method especially with multiple experts	最全面的方法，尤其是多位专家
Not viable for large collections	对大型馆藏不可行

Building a Test Collection 2
构建测试集合

Collect documents	收集文件
Collect queries	收集查询
Submit each query to several different IR systems	将每个查询提交给几个不同的IR系统
Create an empty pool for each query	为每个查询创建一个空池
Place the first *n* documents returned by each system for a query into its pool	将每个系统返回的前n个文档放入其池中以进行查询
Experts study the documents each pool judging which documents are really relevant	专家研究每个池的文件，判断哪些文件真正相关

Building a Test Collection 2 - cont.
构建测试集合

Comments

This approach is called the “*Pooling Method*”

这种方法称为“池化方法”

This method is used at TREC

该方法用于TREC

The “Pools” can still be large.

“池”仍然很大。

At TREC, 100-200 documents per system are placed in it for each query!

在TREC，每个系统都有100-200个文档用于每个查询！

Assumes the IR systems collectively retrieve anything which could be relevant

假设IR系统共同检索任何可能相关的东西

Building a Test Collection 3
构建测试集合

Collect documents

收集文件

Choose documents from the collection which are somehow 'unique'

从集合中选择某些“独特”的文档

Compose queries for each such “*seed*” document

撰写每个此类“种子”文档的查询

Comments

评论

Called the “*Seed Method*”

被称为“种子方法”

Can produce data without too much work

没有太多工作就可以生成数据

Building a Test Collection 3 - cont.
构建测试集合

Comments continued...

Documents on a very unusual topic could be unique

Or, documents which contain keywords not used elsewhere could be unique – these keywords can be used in the artificial queries

Not strictly comparable to general IR: Users would normally expect more than one answer to a query

评论继续......

关于非常不寻常的主题的文件可能是独特的

或者，包含其他地方未使用的关键字的文档可能是唯一的 - 这些关键字可用于人工查询

与一般IR不具有严格的可比性：用户通常期望查询不止一个答案

Building a Test Collection 4
构建测试集合

Collect newspaper documents on a daily basis, starting from some date D_1

每天收集报纸文件，
从某个日期D1开始

Find a unique unpredicted event (e.g. an earthquake) occurring at time D_2

找到一个独特的不可预测的事件（例如地震）发生在D2时刻

Compose a query based on this event

根据此事件撰写查询

Wait four days, until time D_3

等四天，直到时间D3

Search all documents between D_2 and D_3 and judge by hand whether they are relevant

搜索D2和D3之间的所有文档
手工判断它们是否相关

Building a Test Collection 4 - cont.
构建测试集合

Comments

Documents between D_1 and D_2 are by definition not relevant

The search collection for the query is all documents between D_1 and D_3

Documents after D_3 are not part of the search collection, so it does not matter whether they are relevant

根据定义，D1和D2之间的文档不相关

查询的搜索集合是D1和D3之间的所有文档

D3之后的文档不是搜索集合的一部分，因此它们是否相关无关紧要

Building a Test Collection 4 - cont.
构建测试集合

Comments continued...

Called “Páraic Sheridan” Method	被称为 “PáraicSheridan”方法
Ingenious, Scales	巧妙，鳞片
Only works for news data	仅适用于新闻数据

Building a Test Collection 4 - cont.
构建测试集合

Comments continued...

Document collection for each query is different!

- it depends on the date of the unique event.

The closer the date of the event to D_1, the smaller the document collection for queries based on it and vice versa!

Unique events (mostly disasters) are not representative of general documents.

每个查询的文档集合是不同的！

- 这取决于唯一事件的日期。

事件日期越接近D1，基于它的查询文档集越小，反之亦然！

独特事件（主要是灾难）不代表一般文件。

How to Evaluate (‘Offline’ Evaluation)
如何评估（'离线'评估）

Build system	建立系统
Get test queries	获取测试查询
Work out ‘right answers’	找出'正确的答案'
Run queries through the system, store results	通过系统运行查询，存储结果
Compute P, R, MAP, MRR etc	计算P，R，MAP，MRR等