Topic 5: Evaluation

COMP3009J: Information Retrieval

Dr. David Lillis (david.lillis@ucd.ie)

UCD School of Computer Science
Beijing Dublin International College

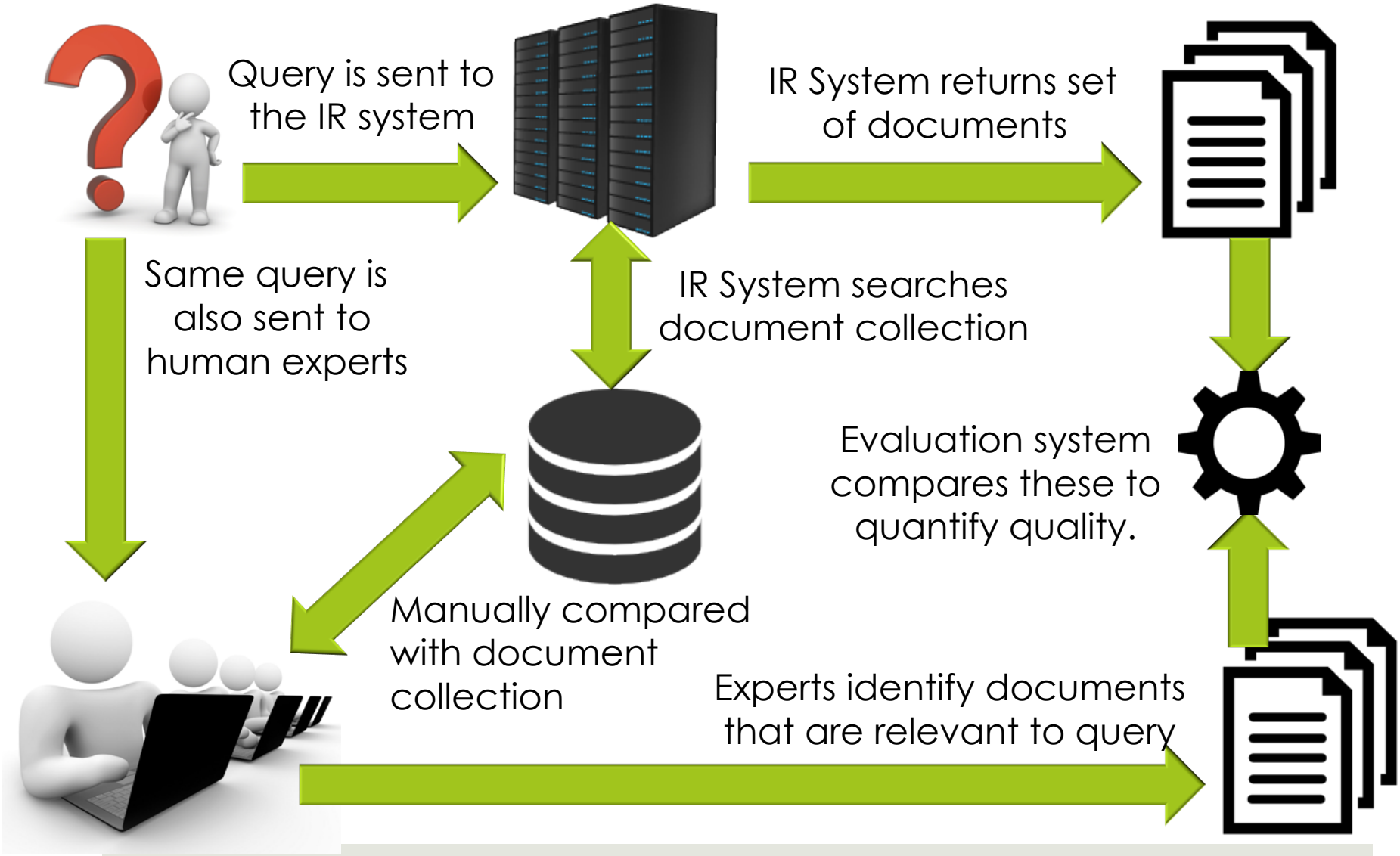
Introduction

- Evaluation is concerned with the question "How well does the system work?"
- There are many measureable quantities for this:
  1. **Processing:** How quickly does the user receive a response? How well are resources utilised?
  2. **User Experience**: Does the user enjoy using the system?
  3. **Search:** How effective is the system in satisfying the user's information need?
- Question 3 is of most interest in this lecture. This evaluate the actual retrieval algorithms that we are using.

Introduction

- Evaluation of the effectiveness of an IR system (particularly in the research area) is a vital topic.
- There are many different techniques used in IR and there needs to be accepted ways to quantify their performance.
- Many metrics exist to do this.
- We will look at the following commonly used metrics:
  - Precision/Recall
  - Precision @ n/R-precision
  - Mean Average Precision (MAP)
  - bPref
  - NDCG

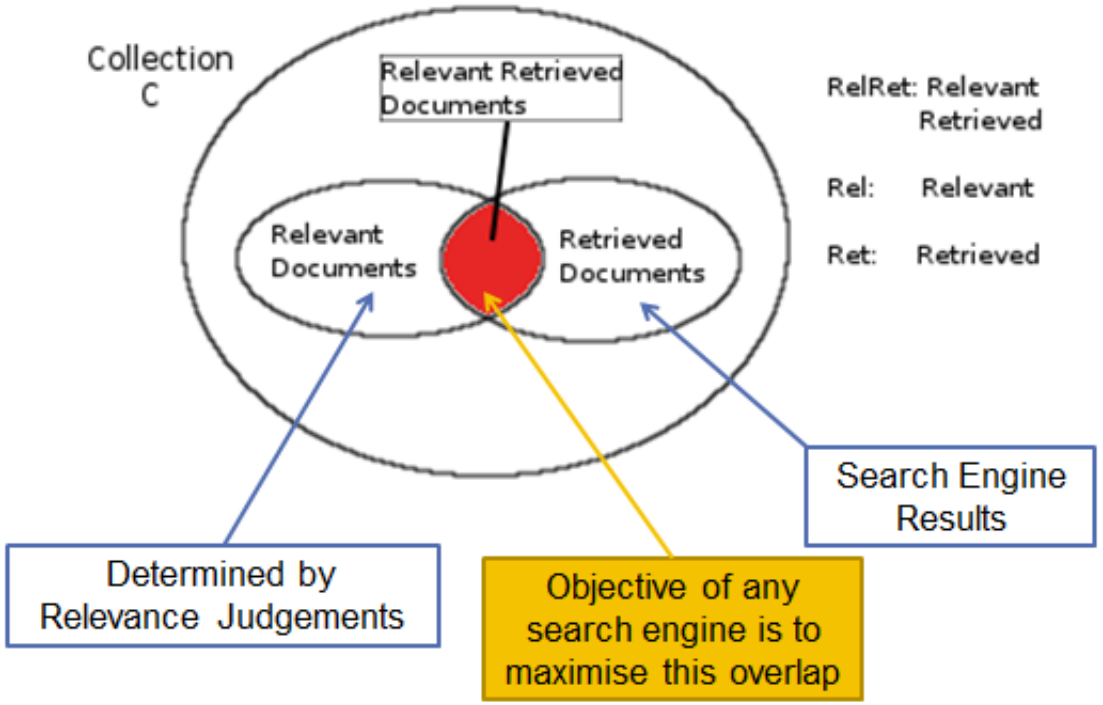
Cranfield Paradigm



Introduction: The Cranfield Paradigm

- A **relevant document** is one that (at least partially) satisfies a user's information need.
- Unfortunately, an information need is a very subjective thing.
- As an alternative, we use experts to judge whether each document is relevant to each query (the Cranfield experiments used aeronautical engineers).
- We can talk about 3 **sets** of documents.
  - The **collection** is the set of all available documents.
  - The **answer set** is the set of documents that an IR system has returned in response to a query.
  - The **relevant set** is the set of documents that have been judged by the experts to be relevant for that query.

Relevance


Collection
C
Relevant Retrieved
Documents
Relevant
Documents
Retrieved
Documents
RelRet: Relevant
Retrieved
Rel: Relevant
Ret: Retrieved
Search Engine
Results
Determined by
Relevance Judgements
Objective of any
search engine is to
maximise this overlap

Introduction

- In reality, the answer set is normally not really a set.
- Generally it is in the form of a **ranked list**.
  - The Boolean Model is the exception to this.
- The purpose of evaluating the effectiveness of an IR technique is to evaluate the quality of this ranked list.

Example (from Modern Information Retrieval)

- Consider a query *q*, on a document collection C where |C|=800
- *Rel* = {d_3, d_5, d_9, d_{25}, d_{39}, d_{44}, d_{56}, d_{71}, d_{89}, d_{123}}
- The ranked list of retrieved documents, *Ret* is given by:

1.	d_{123}	6.	d_9	11.	d_{38}
2.	d_{84}	7.	d_{511}	12.	d_{48}
3.	d_{56}	8.	d_{129}	13.	d_{250}
4.	d_6	9.	d_{187}	14.	d_{113}
5.	d_8	10.	d_{25}	15.	d_3

Precision & Recall

Precision / Recall

- **Precision** and **Recall** are the two most basic (and most widely used) evaluation metrics in IR, upon which many others are based.
- **Precision** is the fraction of the set of retrieved documents (*Ret*) that are relevant (i.e. that are also in *Rel*)
  - $Precision = \frac{|RelRet|}{|Ret|}$
- **Recall** is the fraction of the relevant documents (*Rel*) that have been retrieved (i.e. they are also in *Ret*).
  - $Recall = \frac{|RelRet|}{|Rel|}$

Example

- Looking at the example, we can see:
  - The number of relevant documents for the query is $|Rel| = 10$
  - The number of retrieved documents for the query is $|Ret| = 15$
  - The number of relevant documents in the retrieved set is $|RelRet| = 5$
- $Precision = \frac{|RelRet|}{|Ret|} = \frac{5}{15} = 0.33$
- $Recall = \frac{|RelRet|}{|Rel|} = \frac{5}{10} = 0.5$

Precision vs. Recall

- The two metrics of precision and recall are often inversely related: as one increases the other decreases.
- Precision will be high whenever a system is good at avoiding non-relevant documents.
  - A system can achieve very high precision by retrieving very few documents.
- Recall will be high whenever a system finds many relevant documents.
  - A system can achieve 100% recall simply by retrieving all the documents in the collection.

Precision vs. Recall

- Which is most important?
- That is task-dependent!
- Example: Users searching the web want **high precision** (i.e. they want the returned documents to be relevant).
- Because of the size of the web, there are very many documents that will help satisfy the information need, so the user does not need to examine all of them (i.e. recall does not need to be high).
- Instead, the user wishes to avoid wasting time looking at non-relevant documents.

Precision vs. Recall

- On the other hand, a patent lawyer researching a patent must ensure that they get all of the relevant documents and hence they want **high recall**.
- If any relevant documents are missed, this may have serious consequences so it is essential that all relevant documents are returned.
- They will tolerate lower precision to facilitate this (i.e. they are more likely to be willing to read through some non-relevant documents).
- Ideally every IR system would have both recall and precision of 100% (the answer set is equal to the relevant set).

Precision vs. Recall

- For older, smaller document collections, the calculation of precision and recall was easy.
- Nowadays, because document collections are so huge, it is very difficult to identify all the relevant documents for every query.
- This makes it often impossible to calculate recall accurately.
- Thus, precision tends to be preferred, in addition to other metrics based on precision.

Problems with Precision

- Precision is a **set-based**, **unranked** retrieval metric (as is recall)
- It is a single-value metric based on the entire list of results that was returned by a search engine.
- It does not assess the manner in which the results are ranked.
- Most IR systems return ranked lists, not sets of documents, so users are most likely to look at the top of the list first.

Problems with Precision

Rank	Engine A	Engine B
1	N	R
2	N	R
3	N	N
4	R	N
5	R	N

- N = Non-relevant
- R = Relevant
- Precision = 40% for both A and B
- **BUT,** clearly Engine B is better, because its top-ranked documents are relevant, as a user would expect.

Single Value Metrics

Single Value Metrics

- Using Precision and Recall requires two metrics to be calculated.
- We would prefer to have a single metric that captures the overall performance of the system:
  - Precision at n (P@n)
  - R-precision
  - Mean Average Precision (MAP)
  - Binary Preference (bpref)
  - Normalised Discounted Cumulated Gain (NDCG)

Precision at *n*

- Sometimes we are interested in the precision amongst the top *n* results.
- This is particularly suitable for web search systems where people typically look no further than the first few results.
- For example, we may be interested in the precision after 10 documents have been retrieved.
  - Known as "**Precision at 10**", or "**P@10**"
- This is a measure of the quality of the results that a user is likely to look at.

Precision at *n*

- P @ 3 = 0.67
  - (i.e. 2 of the top 3 results were relevant)
- P @ 10 = 0.4
  - (i.e. 4 of the top 10 results were relevant)

Rank	Document
1	d_{123} (r)
2	d_{84}
3	d_{56} (r)
4	d_{6}
5	d_{8}
6	d_{9} (r)
7	d_{511}
8	d_{129}
9	d_{187}
10	d_{25} (r)
11	d_{38}
12	d_{48}
13	d_{250}
14	d_{113}
15	d_{3} (r)

R-precision

- A similar metric is **R-precision**, where we are interested in the precision after a particular recall level has been reached.
- For example, we want to have retrieved 30% of relevant documents, at which point we want to measure the precision.
- Once a document in the list brings recall above 30%, the precision is measured to that point, and this is the R-precision figure.
- This is similar to P@*n* except that *n* is not fixed for all queries.

R-precision

- R-Precision @ R = 0.3
- Recall reaches 0.3 at rank 6. (assume there were 10 relevant documents in the collection)
- At this point, precision is 0.5
- R-Precision for R=0.3 is 0.5

Rank	Document
1	d_{123} (r)
2	d_{84}
3	d_{56} (r)
4	d_{6}
5	d_{8}
6	d_{9} (r)
7	d_{511}
8	d_{129}
9	d_{187}
10	d_{25} (r)
11	d_{38}
12	d_{48}
13	d_{250}
14	d_{113}
15	d_{3} (r)

Mean Average Precision (MAP)

Mean Average Precision (MAP)

- Mean Average Precision (MAP) has for many years been the most commonly used metric in IR literature to evaluate the performance of systems.
- It is a single-value metric based on precision.
- Unlike simple precision, it rewards systems that rank relevant documents at the beginning of the results returned.
- Unlike P@10, it continues to examine the later stages of the ranked list, although with lesser weight.

Mean Average Precision (MAP)

- It involves three steps:
  1. Firstly, we must calculate the precision at each recall point (at each rank where a relevant document is found).
  2. The **Average Precision** for this query is found by dividing the sum of these precision calculations by the total number of relevant documents.
  3. For multiple queries, the same procedure must be performed for each. We must calculate the mean of the queries' average precision values, giving us **Mean Average Precision.**

Example

- As before, we assume that there were 10 relevant documents available for retrieval.
- Average Precision (for this query): $\frac{1.0+0.67+0.5+0.4+0.33}{10} = 0.29$
- This is the AP for one query.
- To get the Mean Average Precision (MAP), we calculate the average over all queries.

Rank	Document	
1	d_{123}	$P = {}^{1}/_{1} = 1.00$
2	d_{84}	
3	d_{56}	$P = {}^{2}/_{3} = 0.67$
4	d_{6}	
5	d_{8}	
6	d_{9}	$P = {}^{3}/_{6} = 0.5$
7	d_{511}	
8	d_{129}	
9	d_{187}	
10	d_{25}	$P = {}^{4}/_{10} = 0.4$
11	d_{38}	
12	d_{48}	
13	d_{250}	
14	d_{113}	
15	d_{3}	$P = {}^{5}/_{15} = 0.33$

bPref: Binary Preference

bPref: Binary Preference

- All of the evaluation techniques we have mentioned so far are based on the **Cranfield Paradigm**.
- In this, test **collections** and **queries** are created that have a known set of **relevant documents** associated with them.
- The point is that for each query, **every** document in the collection is judged to be "relevant" or "non-relevant".
- Where this occurs, we say that we have **"complete relevance judgments"**.

bPref: Binary Preference

- With smaller collections, this Cranfield Paradigm is perfect (i.e. complete relevance judgments exist).
- However, as document collections have become larger, complete judgments have become less common and we have **incomplete judgments**. This means that some documents have not been judged so they may or may not be relevant to test queries.
- With large-scale IR collections (such as those based on the web), this complete judgment Is impossible to achieve, with potentially millions of documents to be judged for relevance against hundreds of queries.

bPref: Binary Preference

- For the evaluation metrics we have seen so far, they are simplified by **assuming** that unjudged documents are non-relevant.
- It was noticed that many unjudged documents had the effect of lowering evaluation score.
  - **NOTE:** This does not mean that the retrieval was worse: whether a document is relevant or not is not affected by whether somebody has judged it.
- This is a problem because evaluation scores no longer accurately reflect the effectiveness of retrieval.
- For example, what would the effect on the Average Precision score be in our example of nobody had judged document d_9?

bPref: Binary Preference

- The idea behind **bPref** is that these unjudged documents should not impact so largely on the ealuation score.
  - From the paper: Buckley & Voorhees, "Retrieval Evaluation with Incomplete Information", SIGIR 2004
- bPref ignores documents that have not been judged.
- The only documents considered are those that were **judged relevant** or **judged non-relevant**.
- This is an accepted metric for large-scale IR systems where complete relevance judgments are impossible to achieve.

bPref: Binary Preference

- bPref, for a query with *R* relevant documents is calculated as:
  - $B = \frac{1}{R}\sum_{r \in R} 1 - \frac{|n \; ranked \; higher \; than \; r|}{R}$
    - where *n* is a member of the first *R* judged non-relevant documents.

- In other words:
  - For each relevant document in that was retrieved in the answer set:
    - Count the number of non-relevant documents above it in the result set (this is $|n \; ranked \; higher \; than \; r|$). This cannot be greater than the total number of relevant documents (R)
    - The score for that document is $1 - \frac{|n \; ranked \; higher \; than \; r|}{R}$
  - Average this score over all the relevant documents.

bPref Calculation

- Key:
  - R: Relevant
  - N: Non-Relevant
  - U: Unjudged
- Assume a total of 4 available relevant documents (R=4).
- bPref = (0.75+0.75+0+0)/4 = 0.375

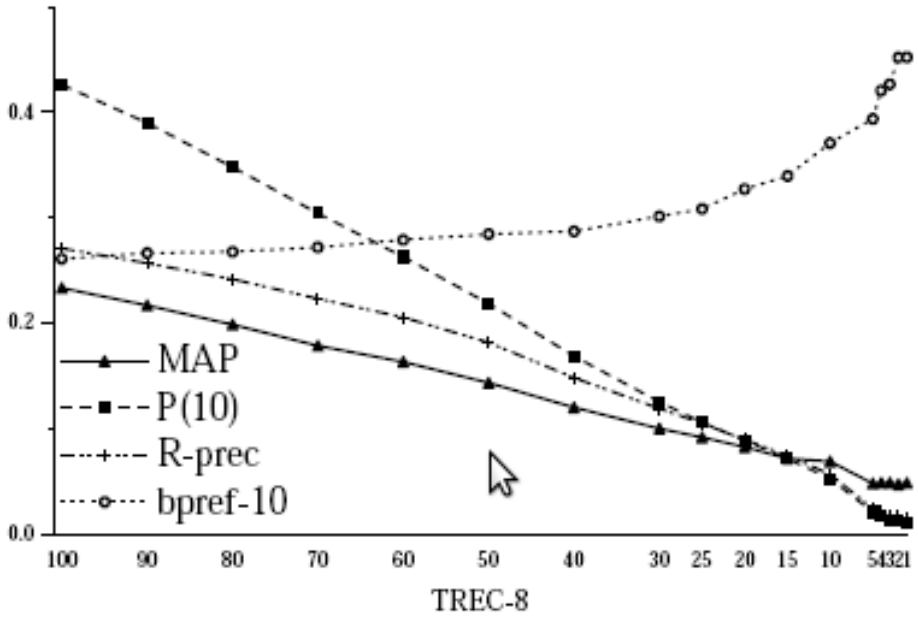
Document	bPref Contribution
N	
R	$1 - {}^{1}/_{4} = 0.75$
U	
R	$1 - {}^{1}/_{4} = 0.75$
U	
N	
N	
N	
R	$1 - {}^{4}/_{4} = 0$
N	
R	$1 - {}^{4}/_{4} = 0$

bPref: Binary Preference

- This works well in most situations.
- However, when *R* is very small (i.e. there are only one or two relevant documents) it fails.
- To overcome this problem, we can instead use bPref-10, which is given by:
  - $bPref10 = \frac{1}{R}\sum_{r \in R} 1 - \frac{|n \; ranked \; higher \; than \; r|}{10+R}$
    - where *n* is a member of the first 10+*R* judged non-relevant documents.

Effect of bPref

- From Buckley and Voorhees 2004 it can be seen that when complete judgments are available, there is no noticeable difference between MAP and bPref-10.
- As the judgments become less complete, bPref is more stable than the others.



Normalised Discounted Cumulated Gain (NDCG)

Normalised Discounted Cumulated Gain (NDCG)

- The metrics so far make use of **binary** relevance judgments:
  - Documents are judged to be **relevant** or **non-relevant**.
  - Any document that helps to satisfy an information need in any way is considered to be **relevant**.
- **BUT:** In reality, some documents are more relevant than others.
  - **Normalised Discounted Cumulated Gain (NDCG)** supports **graded relevance judgments**.

NDCG: Motivation

- NDCG rewards systems that:
  - Rank highly-relevant documents ahead of mildly relevant ones.
  - Position relevant documents in early positions in the ranking.

NDCG: Graded Relevance

- The first thing that is required is a set of **graded relevance judgments**.
  - Let us assume that a 0-3 scale where 3 is a highly relevant document and 0 is a non-relevant document.
  - $R = \{[d_3, 3], [d_5, 3], [d_9, 3], [d_{25}, 2], [d_{39}, 2], [d_{44}, 2], [d_{56}, 1], [d_{71}, 1], [d_{89}, 1], [d_{123}, 1]\}$
    - i.e. d_3, d_5 and d_9 are highly relevant documents.
    - d_{25}, d_{39}, and d_{44} are relevant documents.
    - d_{56}, d_{71}, d_{89} and d_{123} are somewhat relevant documents.
    - Everything else is non-relevant.

NDCG: Example

- Let's revisit the example from before.
- This time, relevant documents are also shown with the degree of relevance attached.
- We create a **gain vector** that records the relevance level at each rank:
- G = (1, 0, 1, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0, 0, 3)
- This is the starting point for our calculations.

Rank	Document	G
1	d_{123} (r: 1)	1
2	d_{84}	0
3	d_{56} (r: 1)	1
4	d_{6}	0
5	d_{8}	0
6	d_{9} (r: 3)	3
7	d_{511}	0
8	d_{129}	0
9	d_{187}	0
10	d_{25} (r: 2)	2
11	d_{38}	0
12	d_{48}	0
13	d_{250}	0
14	d_{113}	0
15	d_{3} (r: 3)	3

Cumulated Gain

- Next, we calculated a **cumulated gain vector** (also sometimes called a "cumulative gain vector").
- Each rank *i* has its own CG value:
  - $CG[i] = \begin{cases} G[1], & i = 1; \\ G[i] + CG[i-1] & i > 1 \end{cases}$
- e.g. at rank 10:
  - CG[10] = G[10] + CG[9]
    = 2 + 5 = 7

Rank	Document	G	CG
1	d_{123} (r: 1)	1	1
2	d_{84}	0	1
3	d_{56} (r: 1)	1	2
4	d_{6}	0	2
5	d_{8}	0	2
6	d_{9} (r: 3)	3	5
7	d_{511}	0	5
8	d_{129}	0	5
9	d_{187}	0	5
10	d_{25} (r: 2)	2	7
11	d_{38}	0	7
12	d_{48}	0	7
13	d_{250}	0	7
14	d_{113}	0	7
15	d_{3} (r: 3)	3	10

Cumulated Gain

- The idea is that each relevant document we find should add to the gain (i.e. the overall usefulness of the list).
- More highly relevant documents add more to the gain than less relevant documents.
- **BUT:** documents late in the list can add as much to the gain as documents early in the list (e.g. at rank 15, a highly-relevant document adds 3 to the cumulated gain: the same as at rank 6).

Rank	Document	G	CG
1	d_{123} (r: 1)	1	1
2	d_{84}	0	1
3	d_{56} (r: 1)	1	2
4	d_{6}	0	2
5	d_{8}	0	2
6	d_{9} (r: 3)	3	5
7	d_{511}	0	5
8	d_{129}	0	5
9	d_{187}	0	5
10	d_{25} (r: 2)	2	7
11	d_{38}	0	7
12	d_{48}	0	7
13	d_{250}	0	7
14	d_{113}	0	7
15	d_{3} (r: 3)	3	10

Cumulated Gain

- This motivates us to create a vector for **Discounted Cumulated Gain (DCG)**.
- Here, later documents add less to the gain than earlier ones.
- It is calculated in the same way as CG, except that the gain at each rank is **discounted** by dividing it by the log of the rank (except for the first rank, which is unaffected).
- Each rank *i* has its own DCG value:
  - $$DCG[i] = \begin{cases} G[1], & i = 1; \\ \frac{G[i]}{log_2 i} + DCG[i-1] & i > 1 \end{cases}$$

Rank	Document	G	CG
1	d_{123} (r: 1)	1	1
2	d_{84}	0	1
3	d_{56} (r: 1)	1	2
4	d_{6}	0	2
5	d_{8}	0	2
6	d_{9} (r: 3)	3	5
7	d_{511}	0	5
8	d_{129}	0	5
9	d_{187}	0	5
10	d_{25} (r: 2)	2	7
11	d_{38}	0	7
12	d_{48}	0	7
13	d_{250}	0	7
14	d_{113}	0	7
15	d_{3} (r: 3)	3	10

DCG

- For this type of data, we could graph the DCG vector to compare two systems.
- However, it is not very suitable in this form.
  - Too many data points
  - Difficult to compare

Rank	Document	G	CG	Calculation	DCG
1	d_{123} (r: 1)	1	1	1	1
2	d_{84}	0	1		1
3	d_{56} (r: 1)	1	2	$\frac{1}{log_2 3} + 1$	1.6
4	d_6	0	2		1.6
5	d_8	0	2		1.6
6	d_9 (r: 3)	3	5	$\frac{3}{log_2 6} + 1.6$	2.8
7	d_{511}	0	5		2.8
8	d_{129}	0	5		2.8
9	d_{187}	0	5		2.8
10	d_{25} (r: 2)	2	7	$\frac{2}{log_2 10} + 2.8$	3.4
11	d_{38}	0	7		3.4
12	d_{48}	0	7		3.4
13	d_{250}	0	7		3.4
14	d_{113}	0	7		3.4
15	d_3 (r: 3)	3	10	$\frac{3}{log_2 15} + 3.4$	4.2

DCG: Analysis

- We have now calculated a Discounted Cumulated Gain vector for a query:

DCG=(1.0, 1.0, 1.6, 1.6, 1.6, 2.8, 2.8, 2.8, 2.8, 3.4, 3.4, 3.4, 3.4, 3.4, 4.2)

- This shows how finding relevant documents increases the quality of the results to that point.
  - More relevant documents contribute more to gain.
  - Relevant documents found earlier in the ranked list also contribute more to gain.

DCG: Analysis

- We have now calculated a Discounted Cumulated Gain vector for a query:

DCG=(1.0, 1.0, 1.6, 1.6, 1.6, 2.8, 2.8, 2.8, 2.8, 3.4, 3.4, 3.4, 3.4, 3.4, 4.2)

- **BUT**: By itself, this is not easy to compare with others.
  - Is 4.2 a good score for this query?
  - Which figure(s) do we choose to compare?
- Most evaluation metrics give a **single value** that is in the range **between 0 and 1**.
- **Normalised** DCG allows us to achieve this.

Normalised DCG (NDCG)

- **Normalised Discounted Cumulated Gain** is calculated by comparing the **DCG** vector against an **Ideal DCG vector**.
- The Ideal DCG vector is the DCG vector that we would see if the IR system had perfect retrieval.
  - i.e. it begins with all the documents of relevance level 3.
    - then it includes all the documents of relevance level 2.
    - then it includes all the documents of relevance level 1.
- We calculate an Ideal DCG vector to be the same length as the DCG vector (with 0 relevance values inserted at the end if there are not enough relevant documents).

NDCG: Example

- Let's look again at the relevance judgments:
- $R = \{[d_3, 3], [d_5, 3], [d_9, 3], [d_{25}, 2], [d_{39}, 2], [d_{44}, 2], [d_{56}, 1], [d_{71}, 1], [d_{89}, 1], [d_{123}, 1]\}$
- For this query, there are:
  - 3 documents at relevance level 3.
  - 3 documents at relevance level 2.
  - 4 documents at relevance level 1.
- The Ideal Gain vector (to compare with a ranked list of length 15) would be:
  - IG = (3, 3, 3, 2, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0)

Rank	G	Calculation	IDCG
1	3	3	3.0
2	3	$\frac{3}{log_2 2} + 3.0$	6.0
3	3	$\frac{3}{log_2 3} + 6.0$	7.9
4	2	$\frac{2}{log_2 4} + 7.9$	8.9
5	2	$\frac{2}{log_2 5} + 8.9$	9.8
6	2	$\frac{2}{log_2 6} + 9.8$	10.5
7	1	$\frac{1}{log_2 7} + 10.5$	10.9
8	1	$\frac{1}{log_2 8} + 10.9$	11.2

- We can calculate the Ideal DCG vector (IDCG) in the same way as for the DCG vector

Rank	G	Calculation	IDCG
9	1	$\frac{1}{log_2 9} + 11.2$	11.5
10	1	$\frac{1}{log_2 10} + 11.5$	11.8
11	0		11.8
12	0		11.8
13	0		11.8
14	0		11.8
15	0		11.8

NDCG: Ideal DCG

- The IDCG vector represents the best possible DCG scores that a perfect IR system would achieve.
- IDCG=(3.0, 6.0, 7.9, 8.9, 9.8, 10.5, 10.9, 11.2, 11.5, 11.8, 11.8, 11.8, 11.8, 11.8, 11.8)
- We can **normalise** the DCG by dividing the score that was actually achieved at each rank by the ideal score, to yield a score between 0 and 1.

NDCG Calculation

Rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DCG	1.0	1.0	1.6	1.6	1.6	2.8	2.8	2.8	2.8	3.4	3.4	3.4	3.4	3.4	4.2
IDCG	3.0	6.0	7.9	8.9	9.8	10.5	10.9	11.2	11.5	11.8	11.8	11.8	11.8	11.8	11.8
NDCG	0.33	0.17	0.20	0.18	0.16	0.27	0.26	0.25	0.24	0.29	0.29	0.29	0.29	0.29	0.36

- We still have the problem that we have 15 different scores for the evaluation.
- This is solved in a similar way to Precision@n: we choose a rank to measure NDCG at.
- NDCG@10 is a very commonly used metric: here it is 0.29

Features of NDCG

- Combines document ranks and graded relevance judgments.
- Single measure of quality at any rank, without needing to know recall.
- NDCG@n only considers relevant documents found to that point: not affected by many relevant documents being found very late.
- Gives less weight to relevant documents found late in the ranking.

Which metric should I use?

- We have looked at numerous different metrics for IR evaluation.
- All have their advantages and disadvantages.
- We need to use an appropriate metric in order to evaluate an IR system's performance.
- How "performance" is defined is dependent on the final use of the system:
  - web search;
  - intranet search;
  - research environment;
  - desktop search;
  - legal search;
  - etc.....

Which metric should I use?

- A general rule is that if complete judgments are available any metric can be used.
- MAP gives a good indication of performance within a single metric as it is averaging the results over multiple queries.
- For collections that do not have complete judgments, bPref is a more suitable metric.
- For tasks such as web search (due to user behaviour), metrics like P@10 might be used.
- If graded relevance judgments are available, NDCG is preferred: this has continually gained in popularity in the last few years.
- In reality, most evaluations use multiple metrics.