MSCC-IRWS
Information Retrieval and Web Search
Advanced Topic 3: Fusion

Jason Farina

Faculty of Computing Science
Griffith College Dublin

jason.farina@gcd.ie

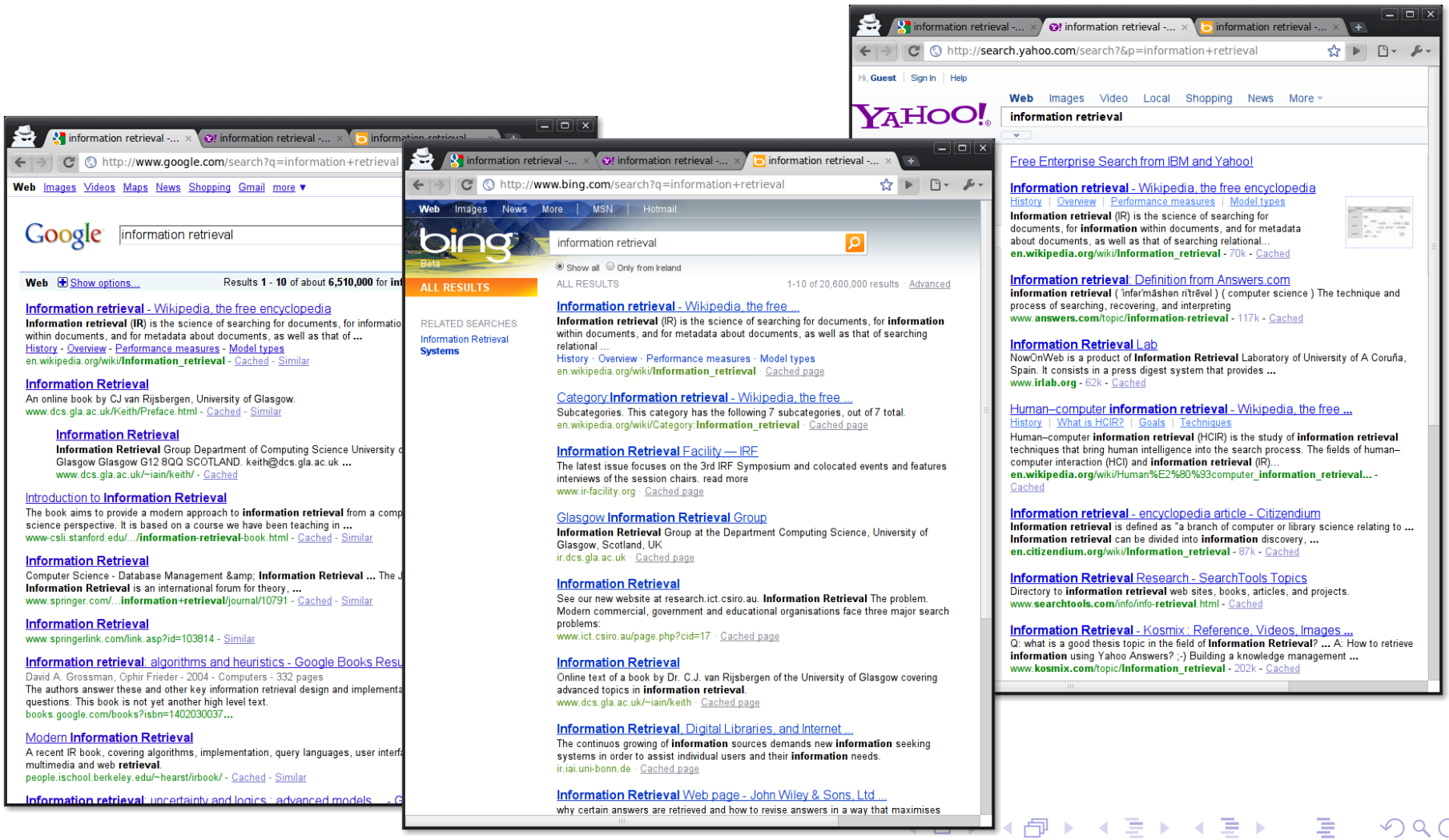
Introduction

- We have already seen how there are many algorithms that can be used for IR.
- In particular, we have looked at the Boolean, Extended Boolean, Vector Space and Probabilistic Models in detail.
- If there was one algorithm that works better than all the others all the time, we would just use that.
- There isn't!

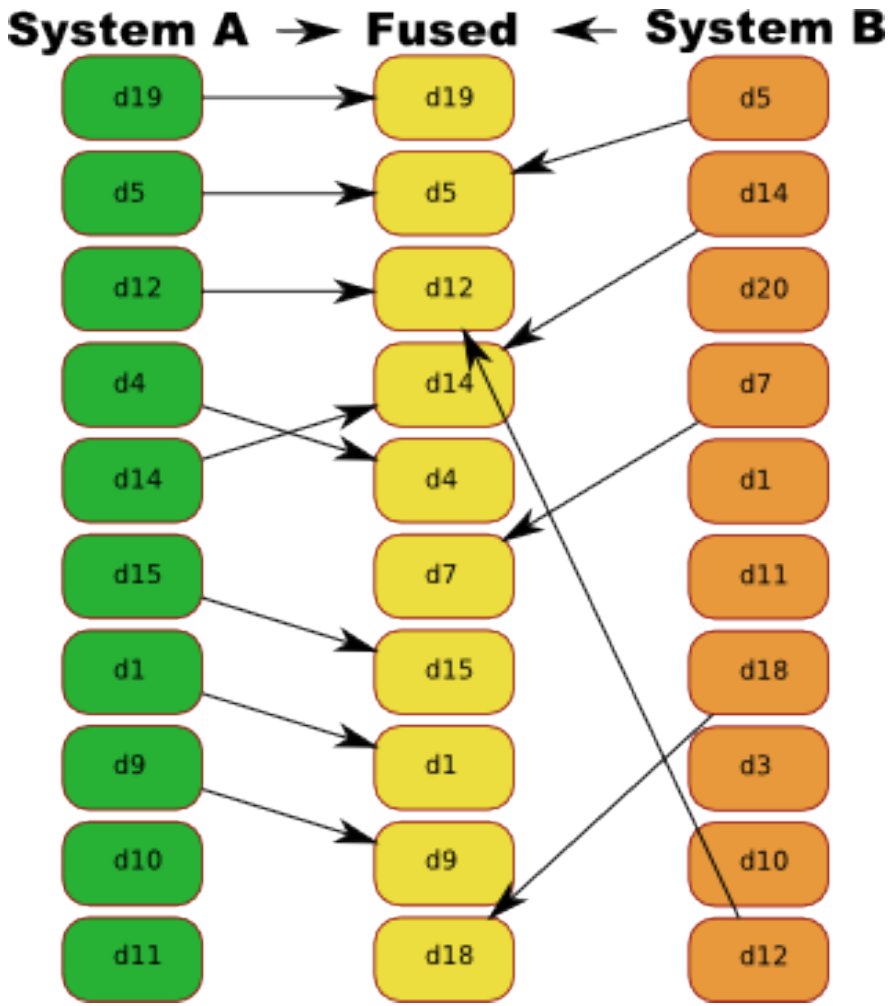
Introduction

- During the 1990s, an area of research called *data fusion*, *collection fusion* or *results merging* became increasingly popular.
- This involves combining the outputs of a number of different IR systems into a single result set that can be shown to a user.
- It is hoped that this combined result set will be of better quality than the individual input result sets.

Introduction



Introduction



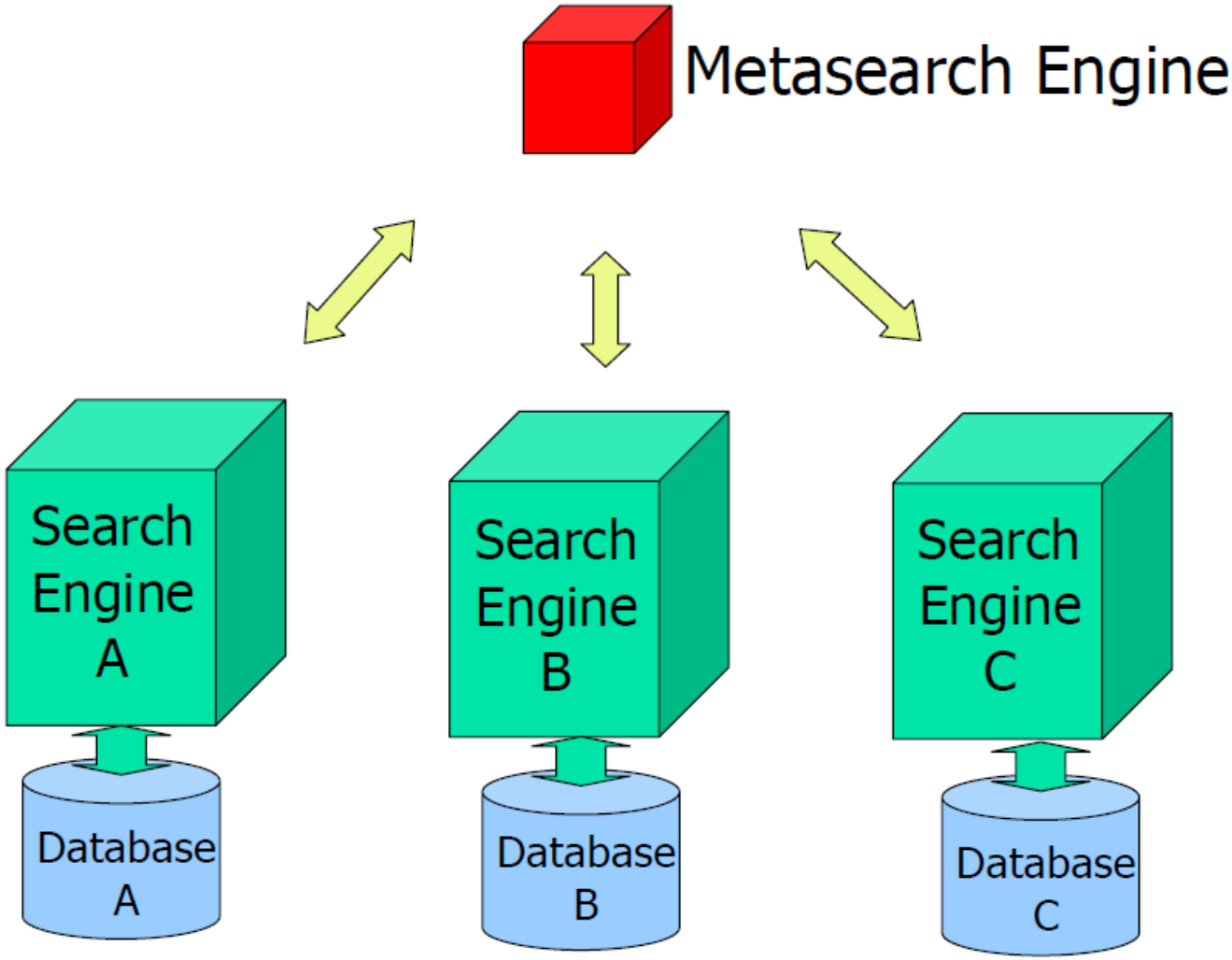
Introduction

- As with most regular Information Retrieval algorithms, we wish to assign some kind of a score to each document.
- Once this is calculated, we rank the documents from the highest score to the lowest.
- Unlike ordinary Information Retrieval, the documents themselves are not taken into account when calculating the score.
- The information that is used comes from the systems that provide the input result sets.

Applications of Fusion

- **Metasearch:** This involves fusion of result sets returned by autonomous, complete search engines.
  - These engines are developed to act alone, so relevance scores are generally not available.
  - Information about the document collection they use is also typically unavailable.
  - Sometimes called *External Metasearch*, to differentiate it from *Internal Metasearch*

Introduction



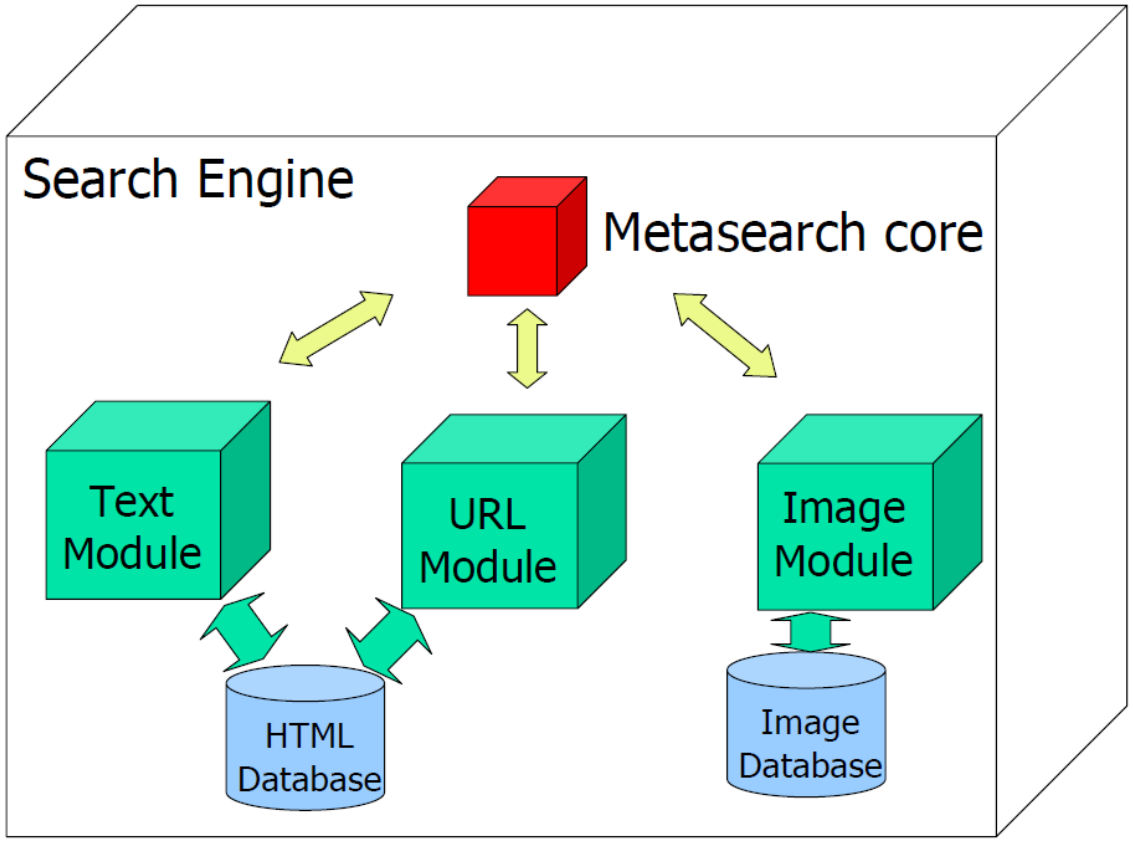
Applications of Fusion

- **Distributed IR:** Numerous IR systems are designed to co-operate with one another.
  - Each has its own separate index and returns results which are then fused.
  - As a co-operative system, relevance scores are generally available.
  - Details about the index each system uses are also available.

Applications of Fusion

- **Internal Metasearch:** Numerous algorithms perform searches on the same document collection
  - As with Distributed IR, relevance scores are generally available.
  - The principal difference between this and Distributed IR is the level of database overlap.

Introduction



Database Overlap

- Before developing a fusion algorithm, it is important to consider how much the document collections used by the systems we wish to use overlap.
- There are three different levels of overlap than can occur between databases.
- The level of overlap will have a significant effect on how we treat the result sets when fusing.

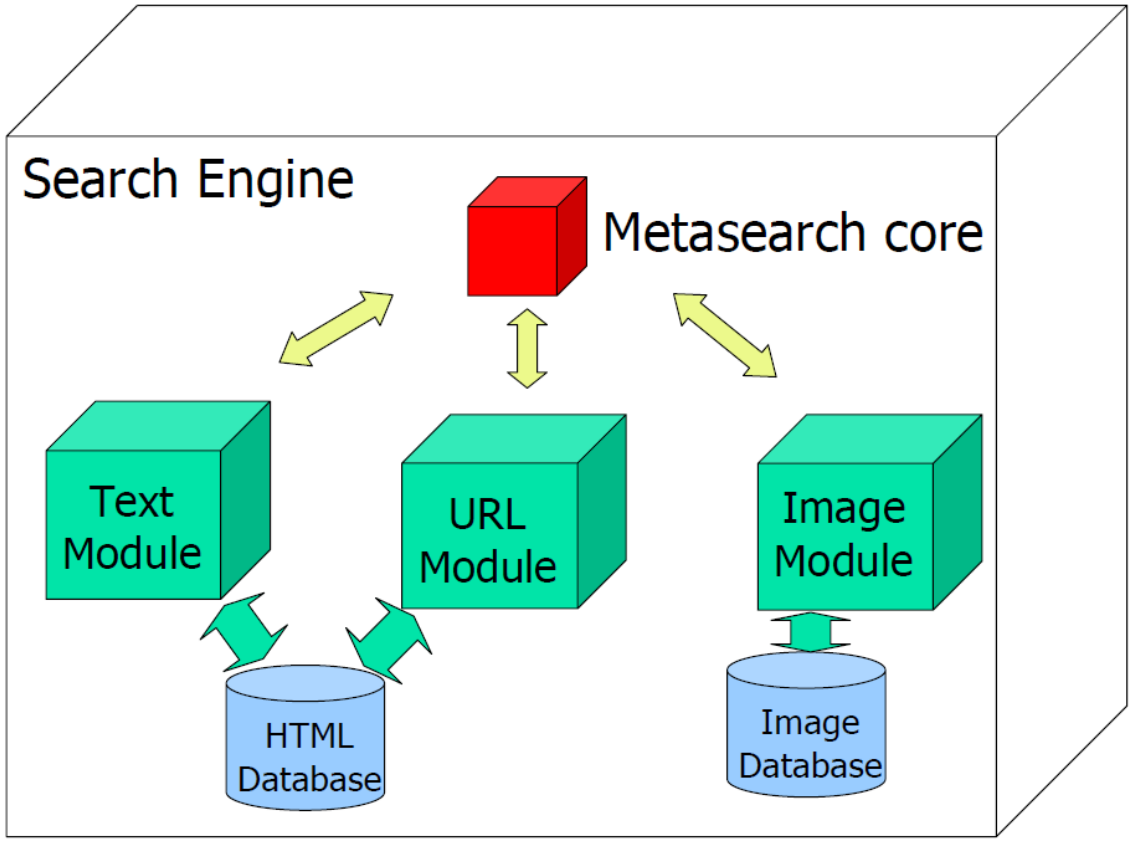
Database Overlap: Disjoint Databases

- Here, the input systems are searching separate, disjoint document collections that have no documents in common.
- A document cannot be returned by more than one input system, since it will not appear in more than one index.
- Distributed IR is frequently implemented by using disjoint databases.
- Fusion of disjoint databases is frequently known as *Collection Fusion*.

Database Overlap: Identical Databases

- The input systems each apply their own IR algorithm to the same set of documents.
- Documents will frequently appear in multiple result sets.
- Appearing in multiple result sets is frequently taken as further evidence that it is relevant.
- This has become known as *Data Fusion* and will be the main focus of this course.
- Internal Metasearch is generally a Data Fusion task.

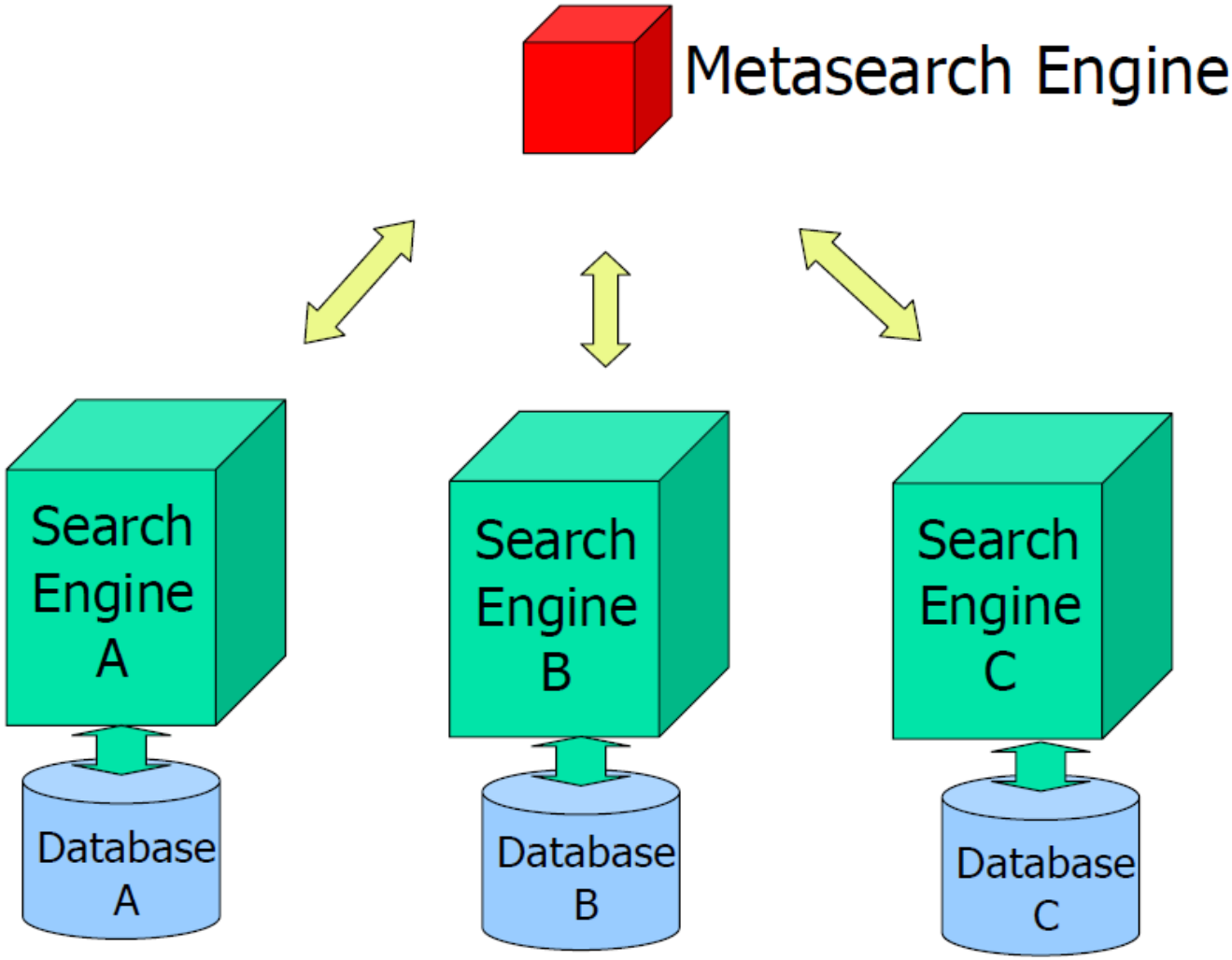
Identical DB



Database Overlap: Overlapping Databases

- The document collections being used by the various input systems have some level of overlap, but are not identical.
- Documents may appear in multiple result sets.
- However, it is difficult to draw reliable conclusions about documents appearing in multiple result sets.
- External Metasearch generally involves overlapping databases.

Overlapping DB



Database Overlap: Overlapping Databases

- If we were to rank documents that appear in multiple result sets higher than those appearing only in one, we must consider the reasons why a document may appear in only one result set and not another:
  - The document may appear in both document collections but is not considered to be relevant by one system. Here, it is correct to rank it below documents that appear in multiple result sets.
  - The document may only be contained in a single document collection, so the other systems do not have the option of returning it. Here, we may be harming our performance if the document is relevant.
- It is often impossible to tell which of these situations has occurred.

Metasearch Engines

dogpile®

web | images | video | news | local | shopping | white pages

Search | Go Fetch!

Search Results From: Google YAHOO!

MetaCrawler™
United Kingdom

Metasearch: [Web | News | Auctions | Search options]

SEARCH

international results | results in English

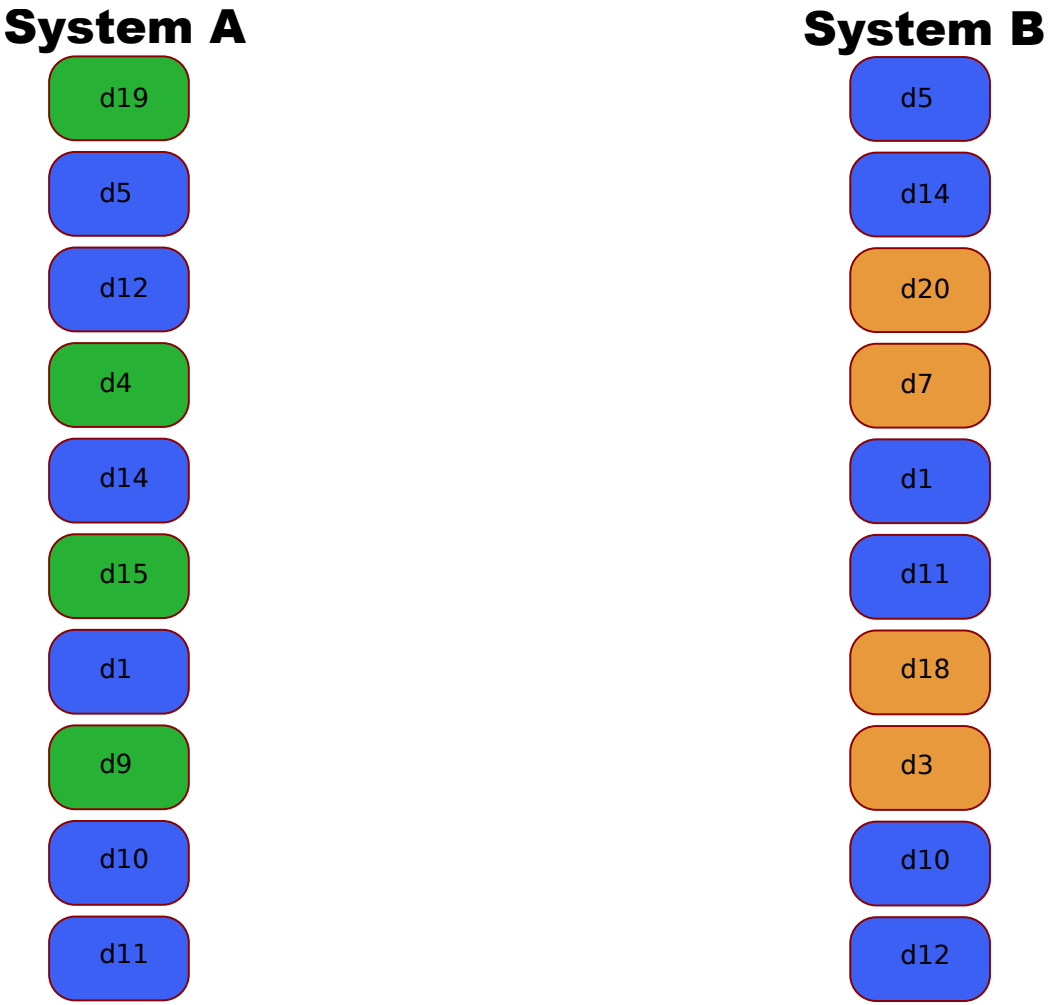
The Skimming Effect

- In general, the most relevant documents in a result set appear at or near the top, when an effective IR algorithm is being used.
- The Skimming Effect argues that "skimming" the top documents from each result set and using these for fusion should give better performance.
- This principle is used by all popular fusion algorithms.

The Chorus Effect

- If multiple input systems agree that a document is relevant, the Chorus Effect argues that this evidence of relevance should be taken into account and that the document should be highly ranked in the fused result set.
- Whether this effect is applicable depends on the level of overlap between the databases used by the input systems.
- For Data Fusion, the Chorus Effect tends to be an important consideration.
- For Collection Fusion, it is not a factor at all, since documents cannot appear in multiple result sets.

The Chorus Effect



The Dark Horse Effect

- The Dark Horse Effect is where one input system returns results of a much different quality than the others.
- This may be as a result of returning either unusually accurate or unusually inaccurate results.
- This contradicts the Chorus Effect, as it would favour identifying the "dark horse" and just using its results, rather than fusing it with others.
- The Dark Horse Effect is very difficult to identify, and so is generally ignored.

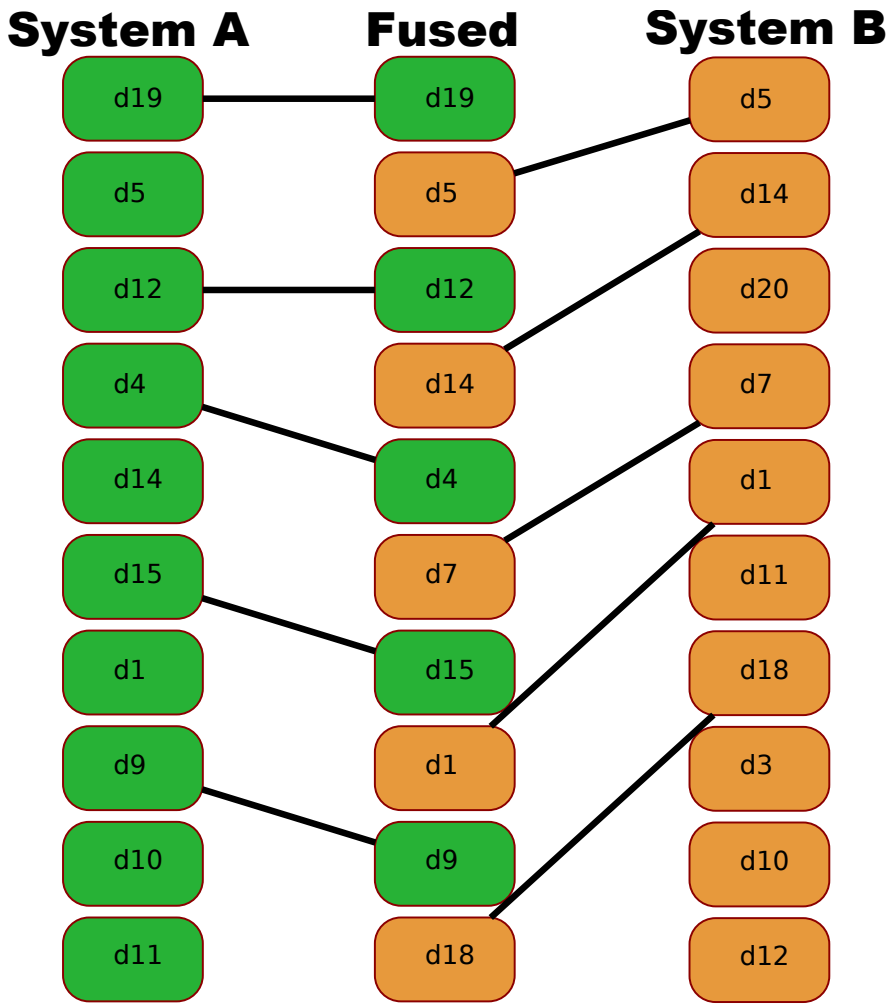
Categories of Data Fusion Techniques

- There are three principal categores of Data Fusion techniques that we will look at:
  - **Rank-Based Fusion:** These examine only the rank that each document occupies in the input result sets. Sometimes necessary if relevance scores are unavailable.
  - **Score-Based Fusion:** The relevance scores can be used as a measurement of how confident an input system is that a document is relevant.
  - **Segment-Based Fusion:** Result sets are divided into groups of documents, rather than examining individual ranks or scores.

Interleaving

- Interleaving is perhaps the simplest Fusion algorithm of all.
- Here, we take one document from the top of each input result set in a "round-robin" fashion and add it to the fused result set.
- The effectiveness of this technique is, however, poor.
- There is an assumption that every result set is of equal quality, which can have the result that the better result sets are diluted by being merged with non-relevant documents from poorer systems.

Interleaving Example



Other Rank-Based Techniques

- A variation on interleaving is to use historical data to estimate which of the input systems tends to perform better.
- A weighted version of interleaving is then used so that more documents are taken from the better systems. Developed by Ellen Voorhees in 1994.
- Another approach is to treat the process as an election where a few electors (the input systems) must choose between many candidates (the documents). This is the approach taken in the *Borda-Fuse* and *Condorcet-Fuse* algorithms, proposed by Aslam and Montague.

CombSUM

- CombSUM is a popular algorithm for performing fusion using the relevance scores that each document is assigned by the input IR system. It was proposed by Fox and Shaw in 1994.
- The final score on which document is ranked in the fused result set is calculated by adding the individual scores it was given in each of the input result sets.
- High scores in the input result sets are carried to the fused result set, so the Skimming Effect is in use.
- Also, because the scores are added, documents that appear in multiple result sets will also receive a boost, exploiting the Chorus Effect.

CombSUM Example

- Example:
  - Document #1 is given a score of 0.45 by System A
  - Document #1 is given a score of 0.3 by System B
  - Document #1 is given a score of 0.35 by System C
  - Document #1's CombSUM score is $0.45 + 0.3 + 0.35 = 1.1$
- Example 2:
  - Document #2 is given a score of 0.55 by System A
  - Document #2 is not returned by System B
  - Document #2 is given a score of 0.65 by System C
  - Document #2's CombSUM score is $0.55 + 0 + 0.65 = 1.2$

Score Normalisation

- There is a difficulty with simply adding raw relevance scores.
- Different IR systems will calculate scores in different ranges:
  - System A: Assigns scores between 0 and 1000
  - System B: Assigns scores between 0 and 1
- We often don't know the theoretical minimum and maximum scores a system *could* produce.
- To get around this, we need to perform *score normalisation* so that each document's score is in a comparable range.

Score Normalisation

- There are a number of approaches to normalising scores.
- The most common is known as *standard normalisation* and is calculated by:

$$normalised\_score = \frac{unnormalised\_score - min\_score}{max\_score - min\_score} \quad (1)$$

- The first-ranked document in each result set will therefore have a score of 1, and the lowest-ranked document will receive a score of 0.
- Once the score of every document has been normalised, we can then apply the CombSUM algorithm.

Score Normalisation

System A	
Document	Score
d_{19}	0.90
d_5	0.85
d_{12}	0.82
d_4	0.79
d_{14}	0.77
d_{15}	0.64
d_1	0.44
d_9	0.43
d_{10}	0.41
d_{11}	0.38

System B	
Document	Score
d_5	943
d_{14}	920
d_{20}	901
d_7	875
d_1	862
d_{11}	811
d_{18}	795
d_3	770
d_{10}	732
d_{12}	712

Fused	
Document	Score
d_5	943.85
d_{14}	920.77
d_{20}	901.00
d_7	875.00
d_1	862.44
d_{11}	811.38
d_{18}	795.00
d_3	770.00
d_{10}	732.41
d_{12}	712.82

Score Normalisation

System A		
Document	Score	Normalised
d_{19}	0.90	1.00
d_5	0.85	0.90
d_{12}	0.82	0.85
d_4	0.79	0.79
d_{14}	0.77	0.75
d_{15}	0.64	0.50
d_1	0.44	0.12
d_9	0.43	0.10
d_{10}	0.41	0.06
d_{11}	0.38	0.00

System B		
Document	Score	Normalised
d_5	943	1.00
d_{14}	920	0.90
d_{20}	901	0.82
d_7	875	0.71
d_1	862	0.65
d_{11}	811	0.43
d_{18}	795	0.36
d_3	770	0.25
d_{10}	732	0.09
d_{12}	712	0.00

Score Normalisation

System A	
Doc	Normalised
d_{19}	1.00
d_5	0.90
d_{12}	0.85
d_4	0.79
d_{14}	0.75
d_{15}	0.50
d_1	0.12
d_9	0.10
d_{10}	0.06
d_{11}	0.00

System B	
Doc	Normalised
d_5	1.00
d_{14}	0.90
d_{20}	0.82
d_7	0.71
d_1	0.65
d_{11}	0.43
d_{18}	0.36
d_3	0.25
d_{10}	0.09
d_{12}	0.00

Fused	
Doc	Score
d_5*	1.90
$d_{14}*$	1.65
d_{19}	1.00
$d_{12}*$	0.85
d_{20}	0.82
d_4	0.79
d_1*	0.77
d_7	0.71
d_{15}	0.50
$d_{11}*$	0.43

CombMNZ: Exploiting the Chorus Effect

- CombMNZ is a variation of CombSUM that has been shown in a variety of scientific studies to give superior results.
- Although CombSUM makes use of the Chorus effect by adding the normalised scores, CombMNZ emphasises this even further.
- It does this by multiplying each document's CombSUM score by the number of result sets it was contained in (i.e. the number of input systems that gave it a non-zero score).
- The MNZ stands for Multiply Non Zero.

CombMNZ Example

- Example:
  - Document #1 is given a score of 0.45 by System A
  - Document #1 is given a score of 0.3 by System B
  - Document #1 is given a score of 0.35 by System C
  - Document #1's CombMNZ score is $(0.45 + 0.3 + 0.35) \times 3 = 3.3$
- Example 2:
  - Document #2 is given a score of 0.55 by System A
  - Document #2 is not returned by System B
  - Document #2 is given a score of 0.65 by System C
  - Document #2's CombMNZ score is $(0.55 + 0 + 0.65) \times 2 = 2.4$

Score Normalisation

System A	
Doc	Normalised
d_{19}	1.00
d_5	0.90
d_{12}	0.85
d_4	0.79
d_{14}	0.75
d_{15}	0.50
d_1	0.12
d_9	0.10
d_{10}	0.06
d_{11}	0.00

System B	
Doc	Normalised
d_5	1.00
d_{14}	0.90
d_{20}	0.82
d_7	0.71
d_1	0.65
d_{11}	0.43
d_{18}	0.36
d_3	0.25
d_{10}	0.09
d_{12}	0.00

Fused	
Doc	Score
d_5*	3.80
$d_{14}*$	3.30
$d_{12}* \uparrow^1$	1.70
$d_1* \uparrow^3$	1.54
d_{19}	1.00
$d_{11}* \uparrow^4$	0.86
d_{20}	0.82
d_4	0.79
d_7	0.71
d_{15}	0.50

CombMNZ: Exploiting the Chorus Effect

- As with CombSUM, CombMNZ can only be used for Data Fusion tasks, since it gives higher rankings to documents that appear in multiple result sets.
- CombMNZ is very simple to calculate and has been shown to be quite effective in practice.
- For this reason, it tends to be the most common baseline technique that researchers compare new algorithms against.

Linear Combination

- One difficulty with CombMNZ and CombSUM is that (like interleaving) every input result set is assumed to be of equal quality.
- The Linear Combination Model addresses this by applying a weighting to each input system.
- The score calculation is similar to CombSUM, except that each document's normalised scores are multiplied by the weighting associated with the input system that returned it before they are added together.

Linear Combination Example

- Example (System A, B and C have weights of 1, 2, 3 respectively):
  - Document #1 is given a score of 0.45 by System A
  - Document #1 is given a score of 0.3 by System B
  - Document #1 is given a score of 0.35 by System C
  - Document #1's overall score is $(0.45 \times 1) + (0.3 \times 2) + (0.35 \times 3) = 2.1$
- Example 2:
  - Document #2 is given a score of 0.55 by System A
  - Document #2 is not returned by System B
  - Document #2 is given a score of 0.65 by System C
  - Document #2's overall score is $(0.55 \times 1) + (0 \times 2) + (0.65 \times 3) = 2.5$

Linear Combination: CORI

- Of course, this introduces another problem: how do we calculate a weighting for each of the input systems?
- The CORI algorithm relies on having access to details about the document collections each input system is using. It is generally used for Distributed IR.
- Here, document collections are given weights in a similar way to how documents are given $tf - idf$ weights.
- Document frequency: number of documents in the document collection that contains a term
- Inverse collection frequency: based on the number of collections that contain the term.

Linear Combination: LMS

- A simpler technique is LMS (using result Length to calculate Merging Score).
- This does not require any prior knowledge of the input systems' document collections.
- It operates on the hypothesis that a system that returns more documents is probably providing better results.
- This was shown to work surprisingly well in practice.

Linear Combination: ProFusion

- The ProFusion metasearch engine used a slightly different version of a Linear Combination.
- Here, the input system's weighting was calculated based on its performance over 25 queries.
- As a metasearch engine, ProFusion had to deal with situations where a document would appear in one underlying document collection but not another.
- They dealt with this by using the maximum score achieved by any document (normalised score multipled by system weighting) to rank it, rather than summing them.
- This meant that documents returned in multiple result sets do not receive a huge boost: just that they have more opportunities to achieve a better score.

Conclusion

Today:

- What is Fusion?
- How does Data Fusion differ from Collection Fusion (database overlap)?
- Rank-Based Fusion techniques (Interleaving)
- Score-Based Fusion techniques (CombSUM, CombMNZ, Linear Combination)

Next Time:

- Segment-Based, Probabilistic Fusion (ProbFuse, SegFuse, SlideFuse).