Fusion: Score-Based

COMP3009J: Information Retrieval

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

UCD School of Computer Science
Beijing Dublin International College

Score-Based Fusion: 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 & Shaw in 1994*.
- The final score on which a 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 set are carried to the fused result set, so the **Skimming Effect** is in use.
- Also, because the scores are added, documents appearing in multiple result sets will also receive a boost, exploiting the **Chorus Effect**.

* Fox, E. A., & Shaw, J. A. (1994). Combination of Multiple Searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2), National Institute of Standards and Technology Special Publication 500-215* (pp. 243–252)

Score-Based: CombSUM Example

- Example 1:
  - 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.
- Consider the situation where different IR systems may 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:

$$normalised\_score = \frac{unnormalised\_score - min\_score}{max\_score - min\_score}$$

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

* Montague, M., & Aslam, J. A. (2001). Relevance score normalization for metasearch. In *CIKM ' 01: Proceedings of the Tenth International Conference on Information and Knowledge Management* (pp. 427–433). New York, NY, USA.

Score Normalisation: CombSUM

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}	0.9	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: CombSUM

System A	
Document	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	
Document	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	
Document	Score
d_5* (*=in both)	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

Score-Based: CombMNZ

- CombMNZ* is a variation of CombSUM that has been shown to give superior results in a variety of scientific studies.
- 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 give it a non-zero score).
- The MNZ stands for "Multiply by Non Zero".

* Fox and Shaw (1994)

Score-Based: CombMNZ Example

- Example 1:
  - 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) x 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) x 2 = 2.4

Score Normalisation: CombMNZ

System A	
Document	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	
Document	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	
Document	Score
d_5* (* = in both)	3.80
d_{14}*	3.30
d_{12}* ($\uparrow^1$)	1.70
d_1* ($\uparrow^3$)	1.53
d_{19}	1.00
d_{11}* ($\uparrow^4$)	0.86
d_{20}	0.82
d_4	0.79
d_7	0.71
d_{15}	0.5

Score-Based: CombMNZ

- As with CombSUM, CombMNZ should 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.

Score-Based: 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 apply 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.

* Vogt & Cottrell (1999)

Linear Combination Example

- Example 1 (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 x 1) + (0.3 x 2) +(0.35 x 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 x 1) + (0 x 2) + (0.65 x 3) = 2.5

Linear Combination: Weights

- Many approaches have been used to calculate weights for a linear combination. Here are two notable ones.
  1. The *CORI* algorithm*, which relies on having access to **details about the document collection** each input system is using (generally used for Distributed IR).
     - **Document Frequency**: number of documents in the document collection that contains a term.
     - **Inverse Collection Freqency**: based on the number of collections that contain the term.
  2. *LMS*** (using result Length to calculate Merging Score): longer result set results in higher weight. Worked surprisingly well in practice.

* Callan, J. P., Lu, Z., & Croft, W. B. (1995). Searching distributed collections with inference networks. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 21–28). New York, NY, USA.

** Rasolofo, Y., Abbaci, F., & Savoy, J. (2001). Approaches to Collection Selection and Results Merging for Distributed Information Retrieval. In *CIKM '01: Proceedings of the Tenth International Conference on Information and Knowledge Management* (pp. 191–198). New York, NY, USA.