Fusion: Case Study: Probability-Based Fusion

COMP3009J: Information Retrieval

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

UCD School of Computer Science
Beijing Dublin International College

ProbFuse - Motivation

- Using a single weight for each system only reflects that the **overall average performance** of one system is better (or worse) than another.
- This can miss some characteristics that we may wish to make use of.
  - An input system may tend to return a few relevant documents at the start of its results, so precision and recall may both be poor.
  - Another system could have good recall, but be poor at ranking its results so as to place relevant documents first.

ProbFuse - Motivation

- A better solution than a single weighting was sought.
- **Probabilistic** data fusion algorithms try to reflect the **positions/ranks** in the result sets where input systems **tend to return relevant documents**.
- **Past results** are analysed during a **training phase** to build a **series of weights**, depending on the position in a result set that a document appears in.

ProbFuse - Overview

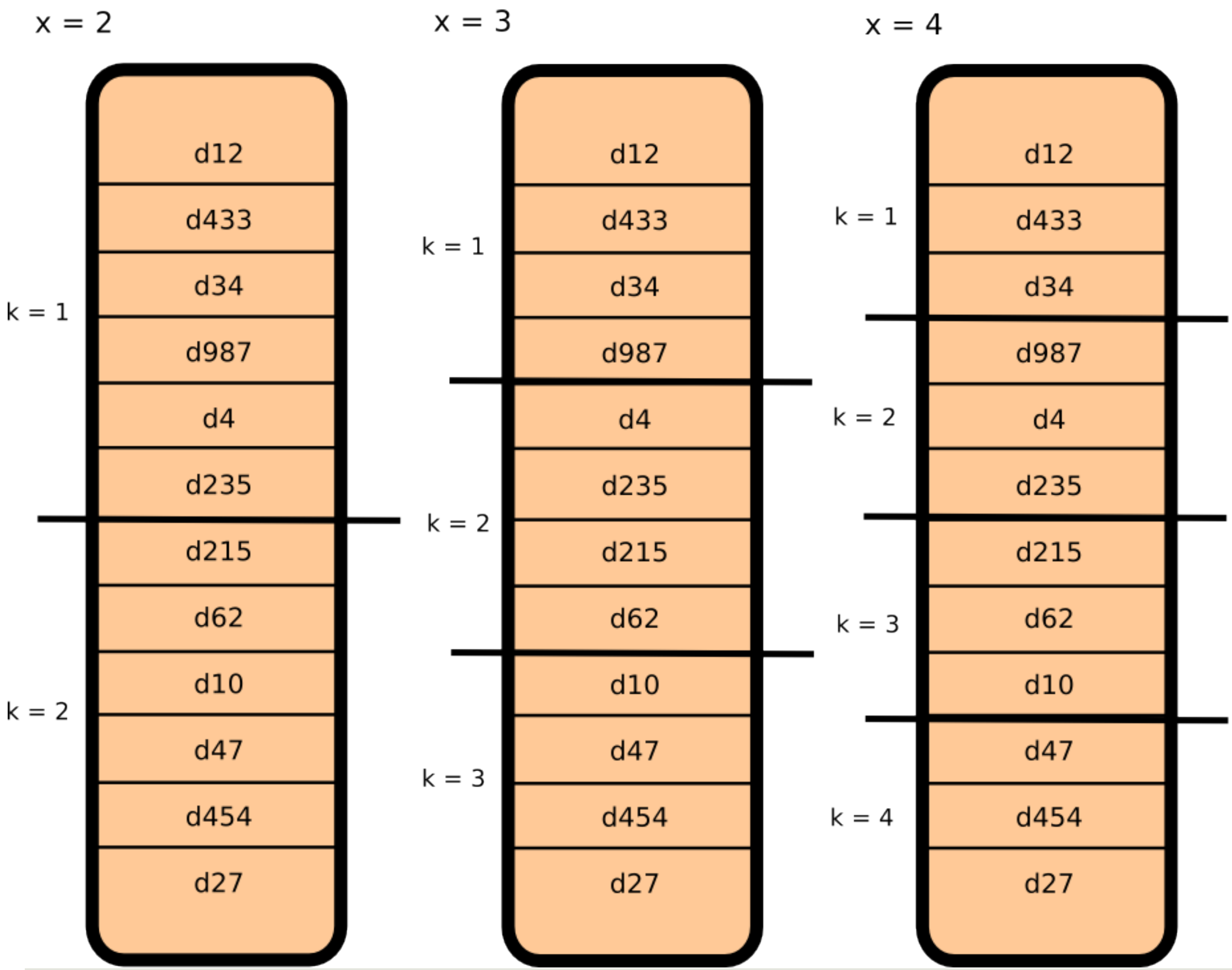
- *ProbFuse* was designed to take this information into account.*
- Each input result set is divided into x **equal-sized segments**.
- The score eventually used to rank a document is based on the **probability that the document is relevant**, given that it was returned by a particular input system in a particular segment.
- These probabilities are calculated by examining **historical data** from each input system.

* Lillis, D., Toolan, F., Collier, R., & Dunnion, J. (2006). ProbFuse: A Probabilistic Approach to Data Fusion. In *Proceedings of the 29th annual international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)* (pp. 139–146). Seattle, WA, USA.

ProbFuse - Overview

- Using ProbFuse for fusion involves **two phases**:
  - **Training Phase:** Probabilities for each input system estimated, based on **historic queries** for which **relevance judgments are available**.
    - Variables:
      - *x*: how many segments to divide the result set into
      - *k*: segment number (k=1 is the first segment, k=2 the second, etc.)
  - **Fusion Phase**: Probabilities used to calculate ranking scores for each document. Used to rank final output result set.

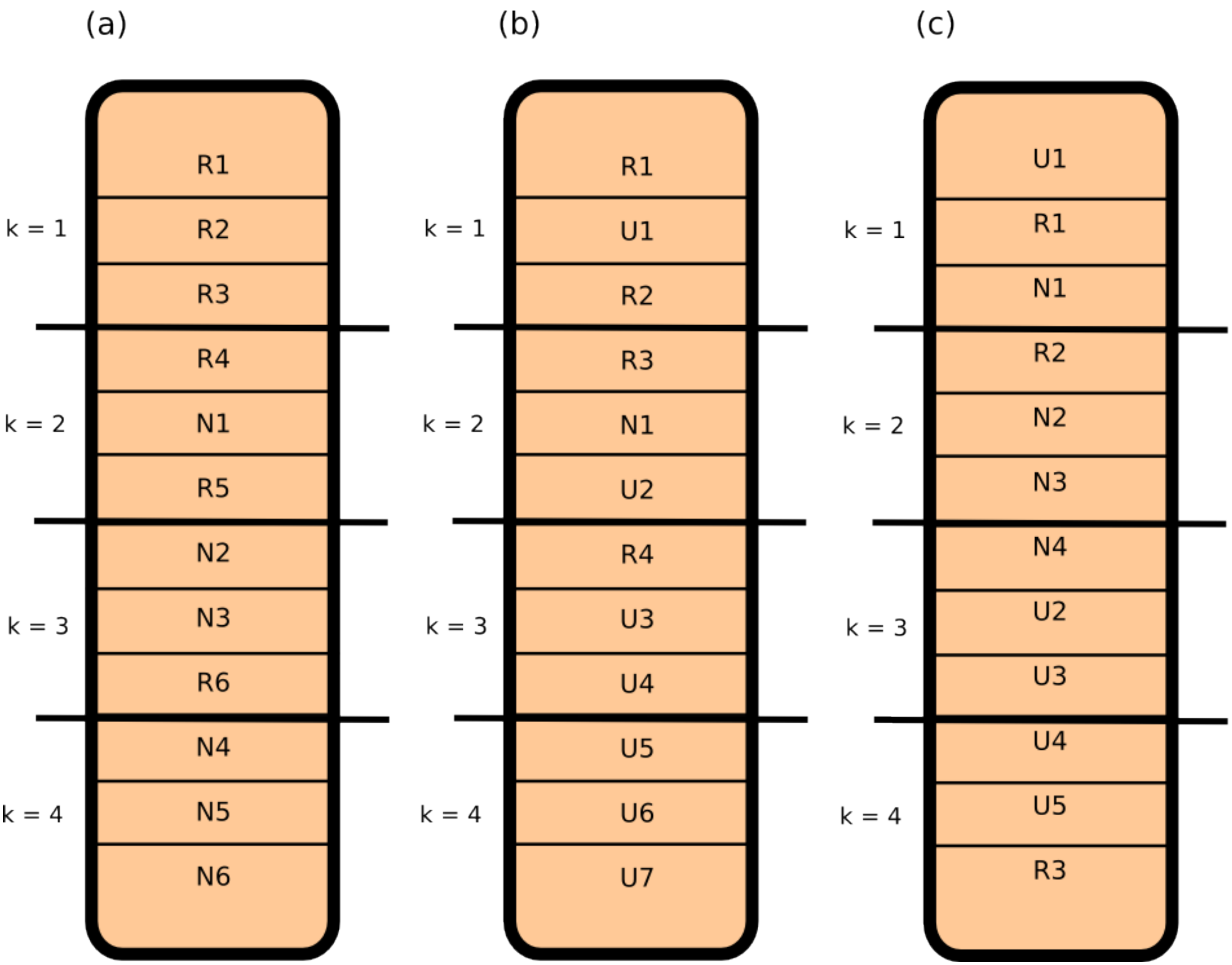
ProbFuse – What are "Segments"?



ProbFuse - Training

- A **set of probabilities** must be calculated for **every input system**.
- For this, a set of **training queries** is required, where relevance judgments are available.
- Calculate the probability that document *d* returned in segment *k* is relevant.
  1. Count all of the relevant documents in segment *k*
  2. Divide by total number of documents in segment *k*
  3. Average over all training queries

ProbFuse - Training



Key:

- R: Relevant
- N: Nonrelevant
- U: Unjudged

$$k = 1 : \frac{\frac{3}{3} + \frac{2}{3} + \frac{1}{3}}{3} = 0.67$$

$$k = 2 : \frac{\frac{2}{3} + \frac{1}{3} + \frac{1}{3}}{3} = 0.44$$

$$k = 3 : \frac{\frac{1}{3} + \frac{1}{3} + \frac{0}{3}}{3} = 0.22$$

$$k = 4 : \frac{\frac{0}{3} + \frac{0}{3} + \frac{1}{3}}{3} = 0.11$$

Result sets returned by the **same input system** in response to **different training queries**

ProbFuse - Fusion

- The score assigned to a document by each underlying input system is the **probability of relevance** associated with the **segment it is returned in**, divided by the **segment number**.
- Dividing by the segment number gives highly-ranked documents an additional boost, so as to **exploit the Skimming Effect**.
- The document's final ranking score is found by **adding each individual score**.
  - This exploits the **Chorus Effect** (higher score from being returned in multiple result sets)
- ProbFuse is therefore only suitable for **data fusion**.

SegFuse - Motivation

- Using ProbFuse, each segment is of **equal size**.
- Experiments on standard datasets showed that dividing result sets into **25 segments** resulted in **improvements over CombMNZ** in MAP and P@10 scores (bpref was inconclusive).
- The input result sets used were up to 1000 documents in length, meaning that **each segment contained 40 documents**.
- This means that document 40 would be treated the same as document 1.

SegFuse - Overview

- Shokouhi argues that this loses information, since relevant documents are more likely to be found in early positions.*
- Two major variations to ProbFuse proposed:
  - Instead of constant segment sizes, the size of the segments **increase exponentially** as we go down the result set.
  - **Normalised scores** used to boost the **Skimming Effect**, rather than dividing by the segment number.
    - For each document, the probability score of its segment was multiplied by its normalised score.

* Shokouhi, M. (2007). Segmentation of Search Engine Results for Effective Data-Fusion. *Advances in Information Retrieval, 4425*

SegFuse - Training

- When training SegFuse, the probability that a document in a given segment is relevant is calculated in **exactly the same way** as for ProbFuse.
- The only difference is that the **size of the segment** varies.
- It is given by: $Size_k = (10 \times 2^{k-1}) - 5$ where k is the segment number.

Segment Number	Size
1	5
2	15
3	35
4	75
5	155

SlideFuse - Motivation

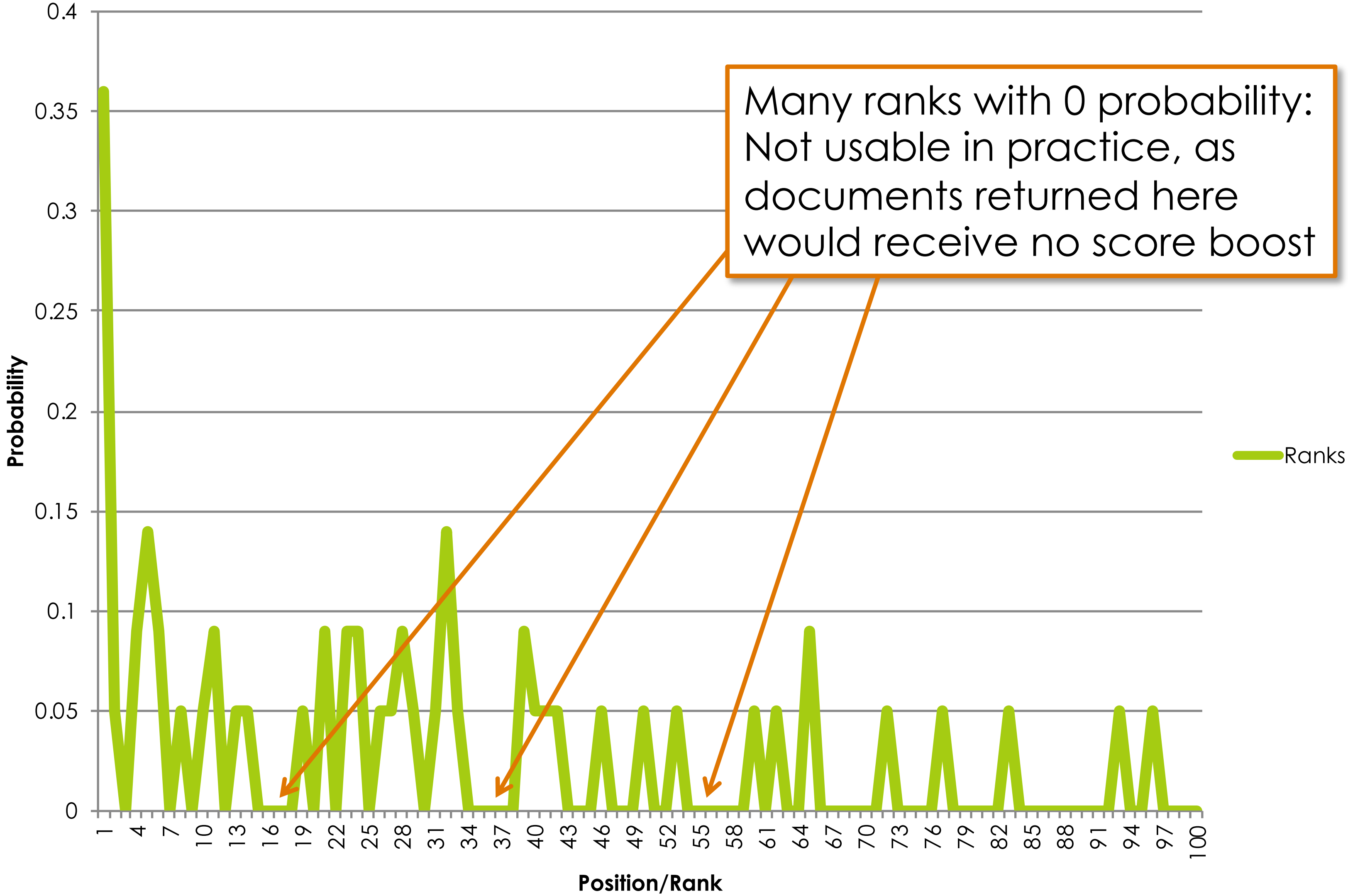
- SegFuse achieved **improvements over ProbFuse** in most of the experiments Shokouhi conducted, especially when measured using MAP.
- However, even with this improved method of dividing result sets into segments, there were still elements of concern.
- In particular, **adjacent documents** could be treated very differently, for instance under SegFuse:
  - Document 20 would be grouped with documents 6-20
  - Document 21 would be grouped with documents 21-55

SlideFuse - Overview

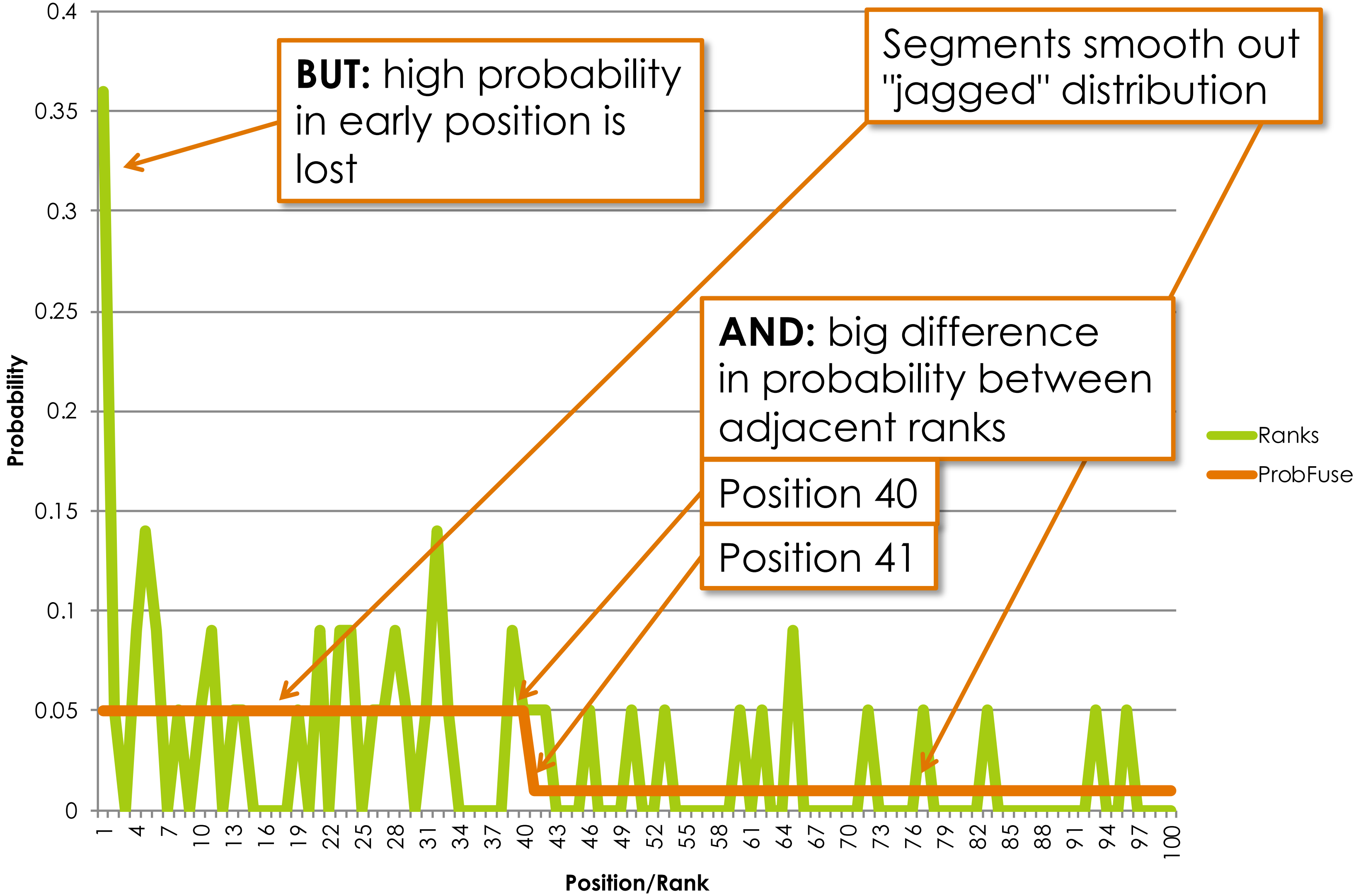
- This cutoff between segments means that documents that are side-by-side in a result set can be **treated very differently**.
- To deal with this, the ***SlideFuse*** algorithm was proposed.*
- **Aside:** Why not just use the probability at **each rank** in the result set?
- Expecially for large-scale tasks, there may be few judged relevant documents available, but we don't want to give a probability of zero to a rank just because no relevant document was returned in that exact rank during training.

* Lillis, D., Toolan, F., Collier, R., & Dunnion, J. (2008). Extending Probabilistic Data Fusion Using Sliding Windows. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, & R. W. White (Eds.), *Advances in Information Retrieval. Proceedings of the 30th European Conference on Information Retrieval Research (ECIR 2008)* (Vol. 4956, pp. 358–369)

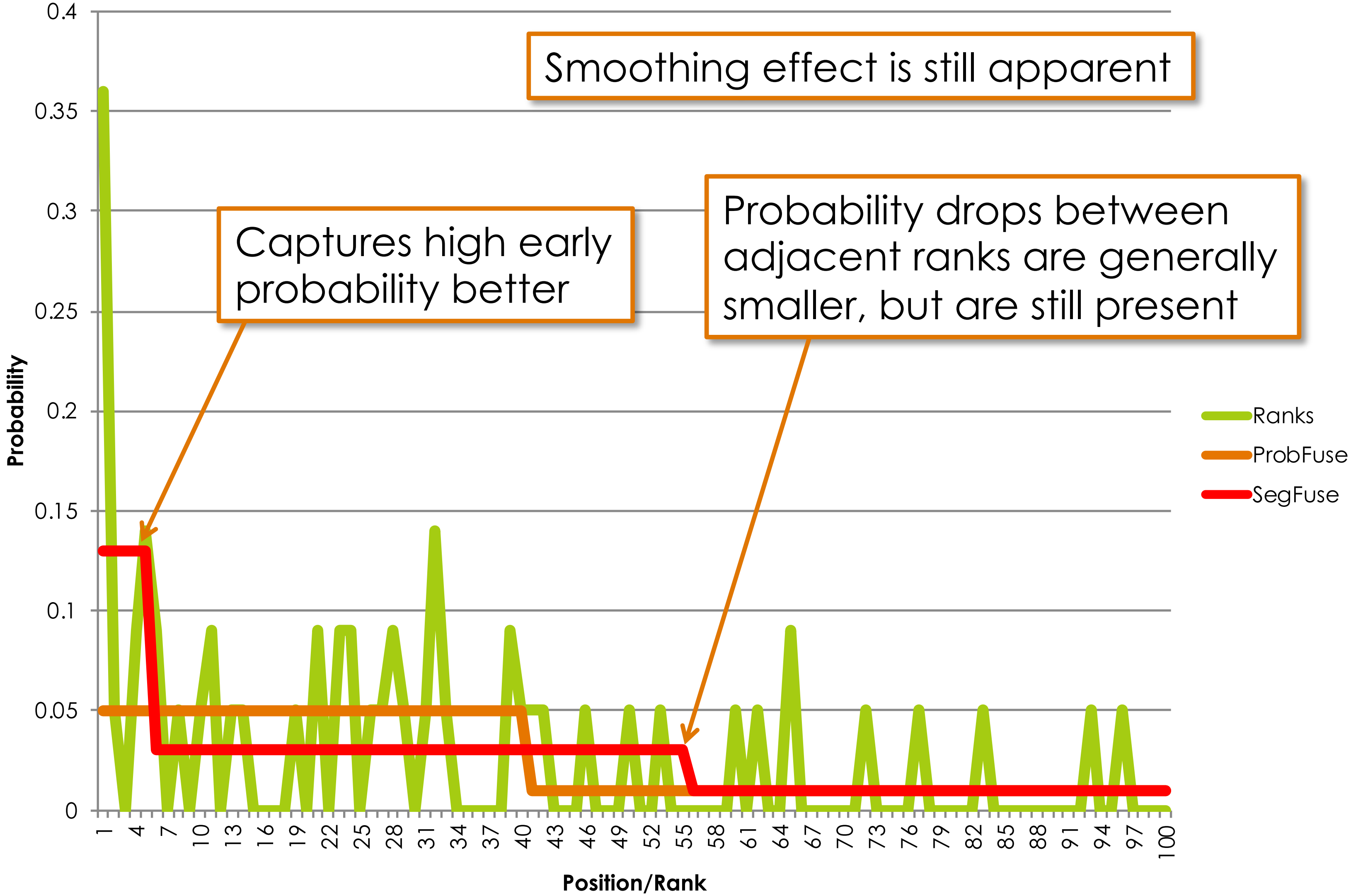
Distribution of Probabilities: Ranks



Distribution of Probabilities: ProbFuse



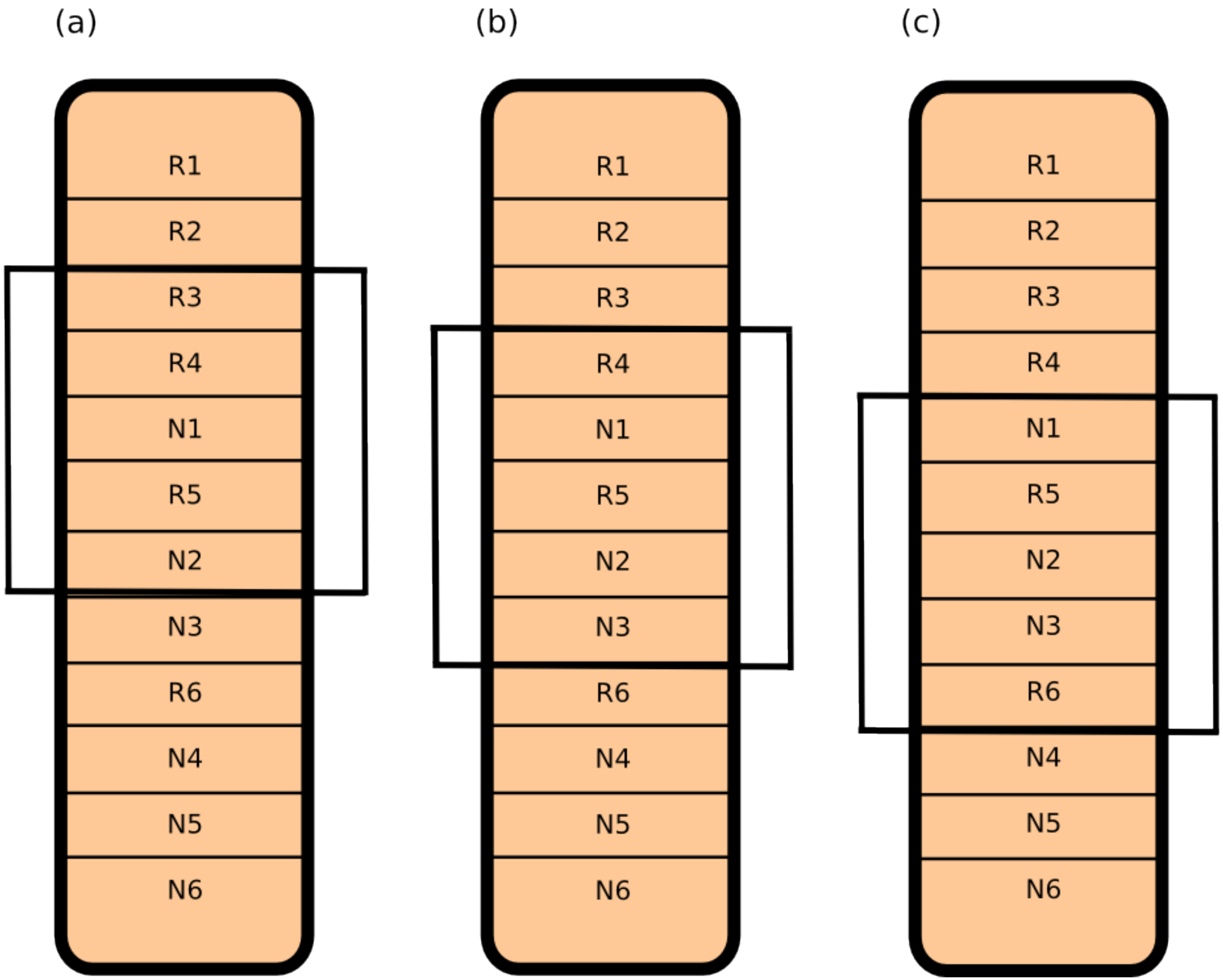
Distribution of Probabilities: SegFuse



SlideFuse - Overview

- Instead of dividing the result sets into segments, we instead use **each rank's neighbours** as evidence in estimating its probability.
- This will allow us to **smooth** the probability distribution graph to avoid the jagged peaks seen in the "ranks-only" distribution and also **avoid the sudden drops** in probability values seen with ProbFuse and SegFuse.
- We describe this as using "**sliding windows**", as the group of documents to be taken into account changes depending on which rank we are examining.

Example: Sliding Windows



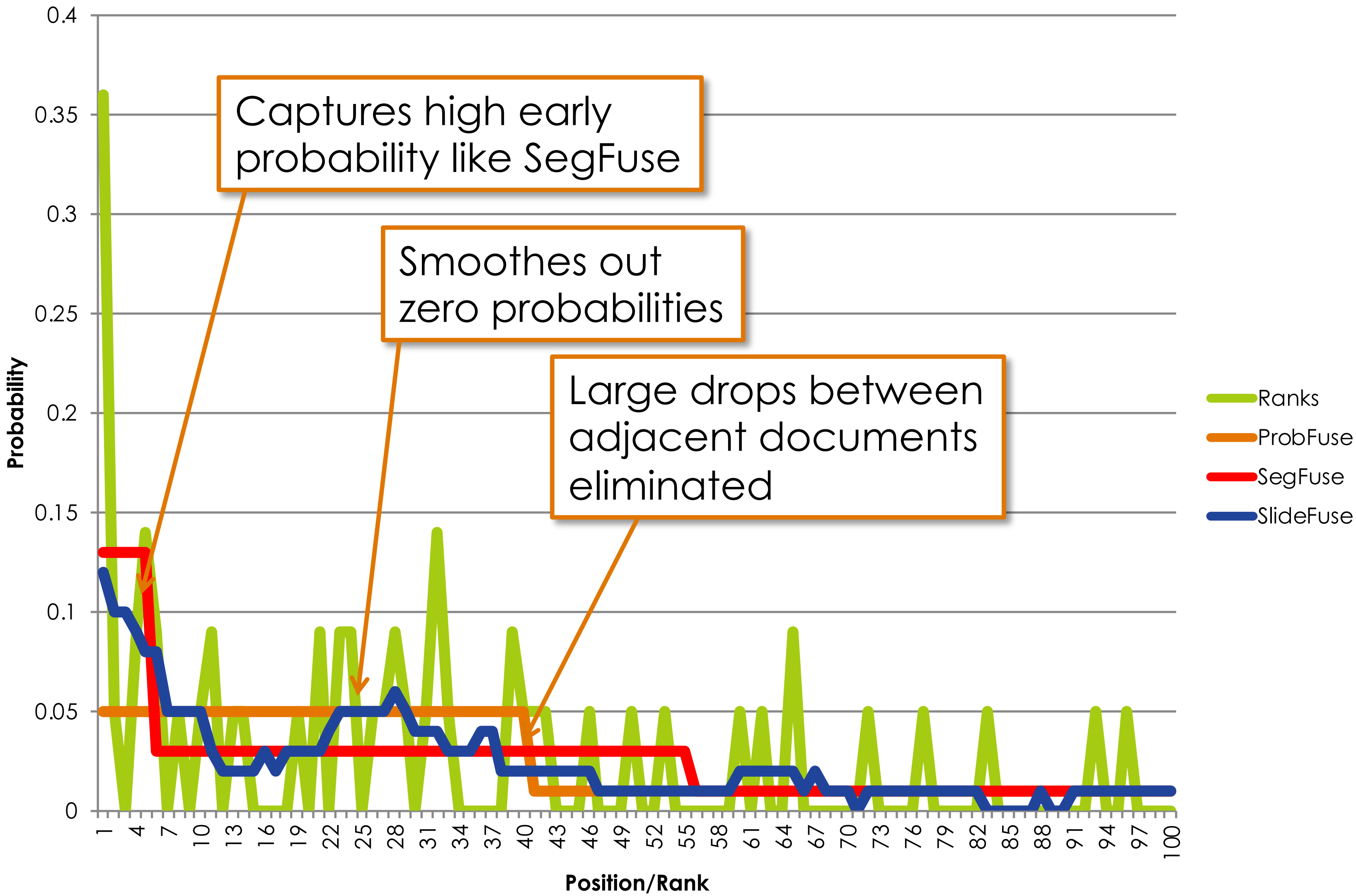
Result set for one training query returned by one system, showing sliding window for document:

a) N1
b) R5
c) N2

SlideFuse - Training

- **Training phase:** for each input system, first calculate the probability of relevance at each rank *r*.
  - Number of relevant documents returned at rank *r* divided by number of training queries.
- **Fusion phase:** final score is the sum of the scores a document receives from the input systems.
  - For each input system, get the average probability in the sliding window that surrounds the document,.

Distribution of Probabilities: SlideFuse



Conclusions

- SlideFuse achieves superior results to CombMNZ, ProbFuse and SegFuse on the TREC-2004 web track data used.
- Statistically significant performance increase for:
  - 4/5 runs measured with MAP
  - 3/5 runs measured with bpref
  - 5/5 runs measured with P@10
- Average difference is < 1% in all other cases