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

Jason Farina

Faculty of Computing Science
Griffith College Dublin
jason.farina@gcd.ie

To Recap...

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)

Today:

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

Data Fusion and Effects

- Recall that there are two *effects* that we try to leverage when developing a Data Fusion algorithm:
  - The *Skimming Effect* favours documents that are returned in early positions of result sets.
  - The *Chorus Effect* favours documents that have been returned by numerous input systems.
- This lecture looks at the process employed in developing data fusion algorithms, examining the motivations behind the approaches taken.
- We recall that *data fusion* specifically refers to systems where each of the inputs are drawn from identical document collections.

ProbFuse - Motivation

- Many of the simpler fusion techniques we have seen do not differentiate between the result sets that they receive as input.
- That is, they assume that each input result set is of equal quality.
  - For simple interleaving, an equal number of documents are taken from each input result set.
  - For CombSum and CombMNZ, every result set is normalised so that the highest-ranked document has a score of 1 and the lowest-ranked has a score of 0.

ProbFuse - Motivation

- Some techniques attempt to take the quality of the input systems into account.
- This is generally done by way of a single weighting score that boosts the better performing input systems.
  - For *Weighted Interleaving*, this weight affects the likelihood that a document will be chosen from a particular result set, with the better result sets providing more documents.
  - In a *Linear Combination*, the normalised score of each document is multiplied by the weight for the input system that returned it.

ProbFuse - Motivation

- Using a single score for each input 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 with 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 could both be poor.
  - Another system could have good recall, but be poor at placing relevant documents at the top of the result set.

ProbFuse - Motivation

- A better solution than a single weighting was sought.
- Rather than one score indicating how good an input system was on average, a group of probabilistic data fusion algorithms also tries to reflect the positions in result sets where input systems tend to return relevant documents.
- This is done by analysing past results of each input system, to build up a series of weights, depending on the position in a result set that a document appears in.

ProbFuse - Overview

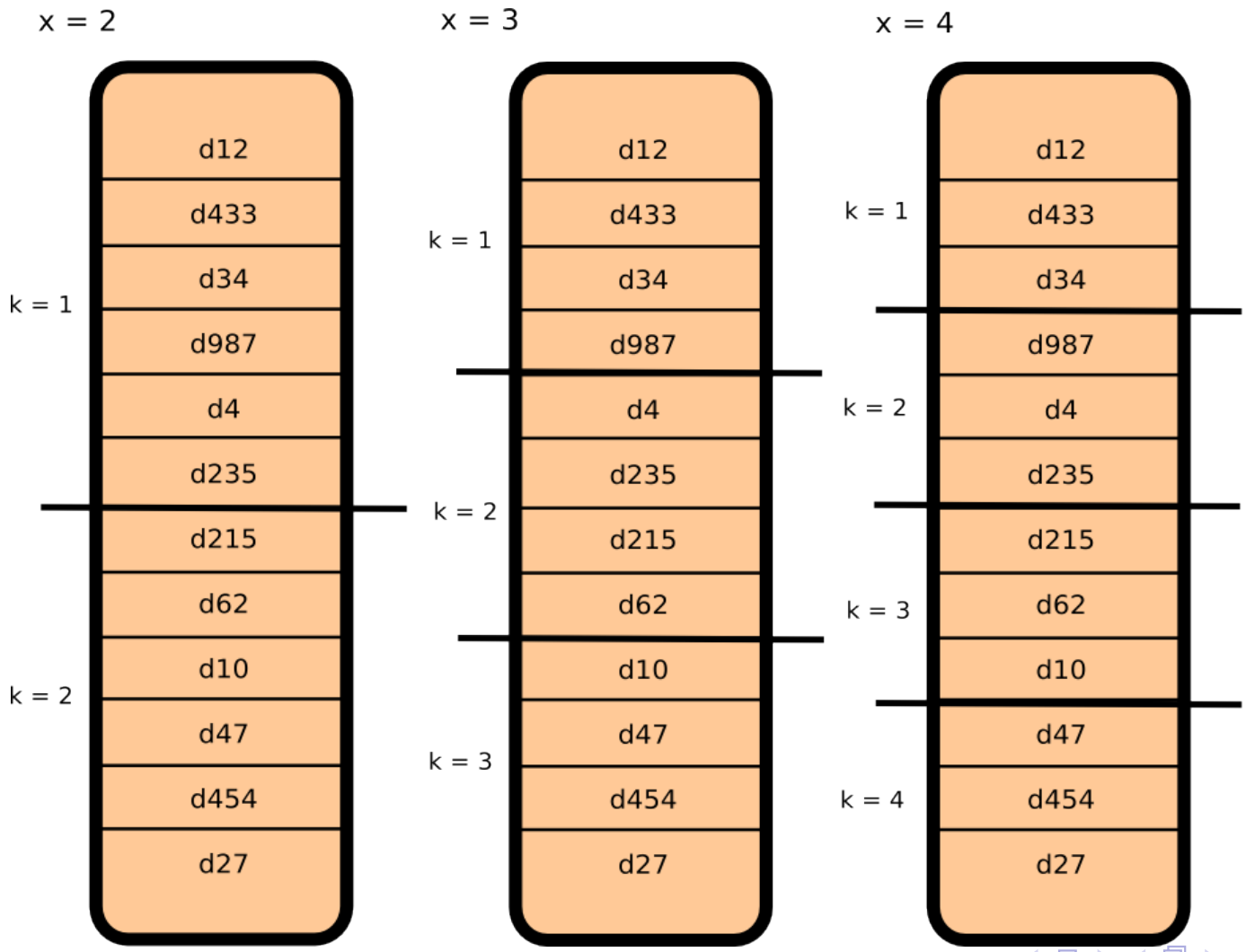
- *ProbFuse* was designed to take this information into account.[1]
- Each input result set is divided into x equal-sized *segments*.
- The scores that are eventually used to rank documents are based on the probability that each document is relevant, given that it was returned by a particular input system in a particular segment.
- These probabilities must be calculated by examining historical data from each input system.

[1] D. Lillis, F. Toolan, R. Collier and J. Dunnion, "ProbFuse: A Probabilistic Approach to Data Fusion", SIGIR 2006

ProbFuse - Overview

- Using ProbFuse for fusion involves two phases:
  1. **Training Phase:** Here, the probabilities for each input system are estimated, based on historic queries for which relevance judgments are available.
     - x: how many segments to divide the result set into
     - k a particular segment ($k = 1$ is the first segment, $k = 2$ the second, etc.).
  2. **Fusion Phase:** Once these probabilities have been estimated, they can be used to calculate ranking scores for each document, which are in turn used to rank the final output result set.

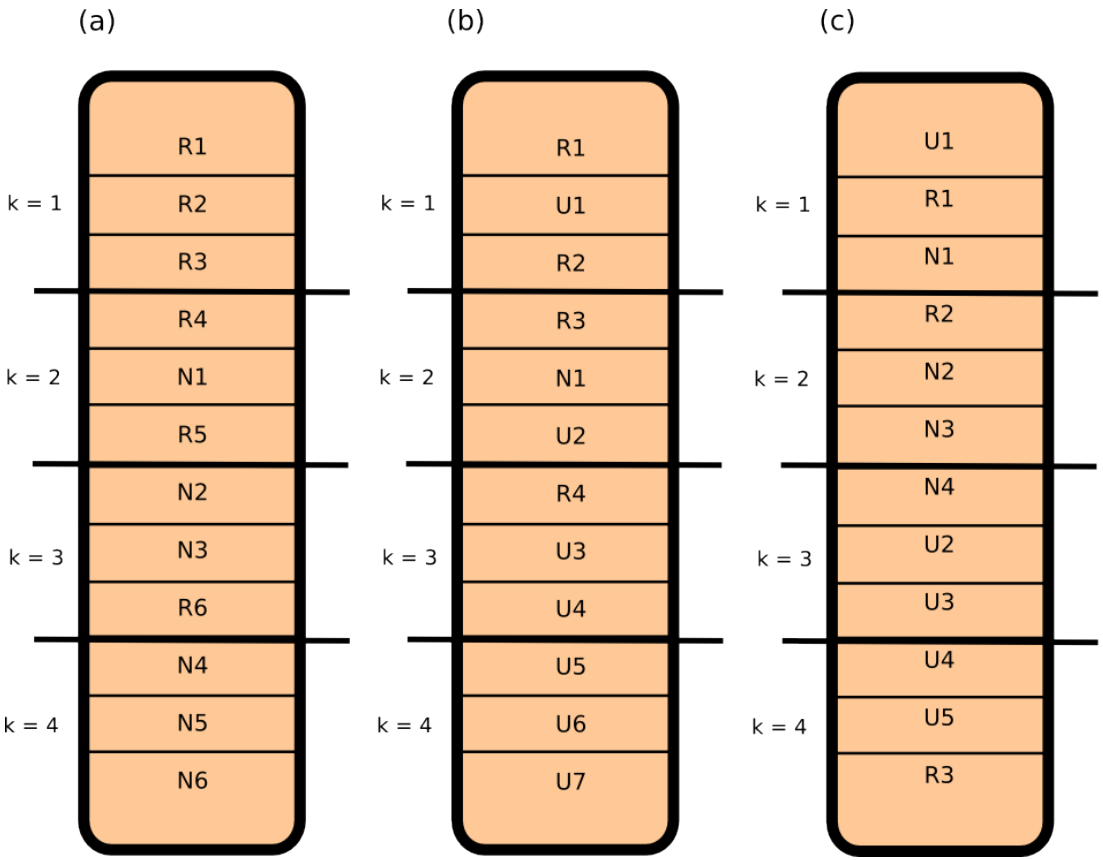
ProbFuse - What are "segments"?



ProbFuse - Training

- A set of probabilities must be calculated for each input system.
- For this, a set of *training queries* is required, where relevance judgments are available (i.e. historical information).
- Probability that document d returned in segment k is relevant.
- This is calculated by dividing the number of documents in segment k that are judged to be relevant by the total number of documents in segment k, and is averaged over all of the training queries.

ProbFuse - Training



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

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

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

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

ProbFuse - Training

- More formally:
- $P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q}$

ProbFuse - Training

- More formally:
- $P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q}$
  - $P(d_k|m)$:= probability that document d returned in segment k is relevant, given that it was returned by retrieval model m.

ProbFuse - Training

- More formally:
- $P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q}$
  - $P(d_k|m)$:= probability that document d returned in segment k is relevant, given that it was returned by retrieval model m.
  - Q := number of training queries

ProbFuse - Training

- More formally:
- $P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q}$
  - $P(d_k|m)$:= probability that document d returned in segment k is relevant, given that it was returned by retrieval model m.
  - Q := number of training queries
  - $|R_{k,q}|$:= number of documents in segment k that are relevant to query q

ProbFuse - Training

- More formally:
- $P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q}$
  - $P(d_k|m)$:= probability that document d returned in segment k is relevant, given that it was returned by retrieval model m.
  - Q := number of training queries
  - $|R_{k,q}|$:= number of documents in segment k that are relevant to query q
  - $|k|$:= number of documents in segment k

ProbFuse - Fusion

- The score assigned to a document by each underlying IR model is the probability of relevance associated with the segment it is returned in, divided by the segment number.
- Dividing by the segment number is to give highly-ranked documents an additional boost, so as to exploit the *Skimming Effect*
- The document's final ranking score found by adding each of these individual scores.
- This rewards documents appearing in multiple result sets and so exploits the *Chorus Effect*
- Because of this, ProbFuse is suitable only for data fusion (i.e. with identical document collections)

ProbFuse - Fusion

- More formally . . .
- $S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k}$

ProbFuse - Fusion

- More formally . . .
- $S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k}$
  - $S_d :=$ ranking score for document d

ProbFuse - Fusion

- More formally . . .
- $S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k}$
  - $S_d :=$ ranking score for document d
  - $M :=$ number of retrieval models

ProbFuse - Fusion

- More formally . . .
- $S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k}$
  - $S_d :=$ ranking score for document d
  - $M :=$ number of retrieval models
  - $P(d_k|m) :=$ probability that document d returned in segment k by retrieval model m is relevant. Assumed to be zero if m does not return d in its result set

ProbFuse - Fusion

- More formally . . .
- $S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k}$
  - $S_d :=$ ranking score for document d
  - $M :=$ number of retrieval models
  - $P(d_k|m) :=$ probability that document d returned in segment k by retrieval model m is relevant. Assumed to be zero if m does not return d in its result set
  - $k :=$ segment that document d appears in (e.g. 1 for the first segment, 2 for second, etc.)

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.
- Shokouhi argues that this loses information, since relevant documents are more likely to be found in early positions.[2]

[2] M. Shokouhi, "Segmentation of Search Engine Results for Effective Data-Fusion", ECIR 2007

SegFuse - Overview

- Shokouhi proposed two major variations to ProbFuse:
  - Firstly, instead of using constant segment sizes, the size of the segment increases exponentially as we go down the result set.
  - Secondly, normalised scores are used to boost the skimming effect, rather than dividing by the segment number.

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

SegFuse - Fusion

- SegFuse calculates the ranking score for a document d (S_d) as follows:
- $S_d = \sum_{m=1}^{M} P(d_k|m) \times (D_m + 1)$

SegFuse - Fusion

- SegFuse calculates the ranking score for a document d (S_d) as follows:
- $S_d = \sum_{m=1}^{M} P(d_k|m) \times (D_m + 1)$
  - $M :=$ number of retrieval models

SegFuse - Fusion

- SegFuse calculates the ranking score for a document d (S_d) as follows:
- $S_d = \sum_{m=1}^{M} P(d_k|m) \times (D_m + 1)$
  - $M :=$ number of retrieval models
  - $P(d_k|m) :=$ probability that document d returned in segment k by retrieval model m is relevant. Assumed to be zero if m does not return d in its result set

SegFuse - Fusion

- SegFuse calculates the ranking score for a document d (S_d) as follows:
- $S_d = \sum_{m=1}^{M} P(d_k|m) \times (D_m + 1)$
  - $M :=$ number of retrieval models
  - $P(d_k|m) :=$ probability that document d returned in segment k by retrieval model m is relevant. Assumed to be zero if m does not return d in its result set
  - $D_m :=$ the normalised score for document d given by retrieval model m

SegFuse - Fusion

- $S_d = \sum_{m=1}^{M} P(d_k|m) \times (D_m + 1)$
- As with ProbFuse, the scores from each input system are added, thus exploiting the *Chorus Effect*
- This also means that SegFuse is also a *data fusion* technique.
- Recall that the normalised score will be 1 for the highest-ranked document and 0 for the lowest.
- The multiplication by $(D_m + 1)$ thus gives an additional boost to documents higher up in the result sets, thus exploiting the *Skimming Effect*.

SlideFuse - Motivation

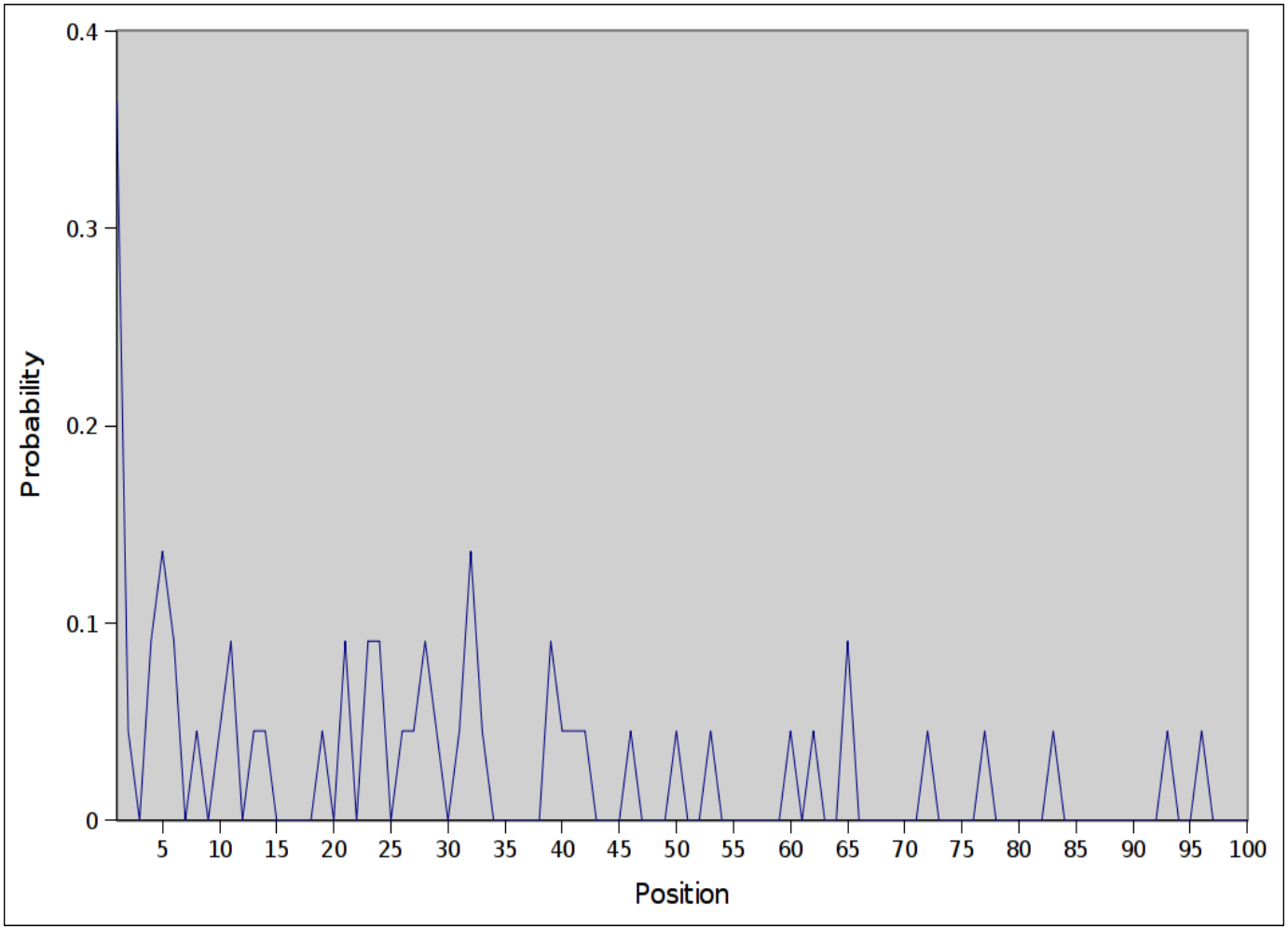
- SegFuse achieved improvements over ProbFuse in most of the experiments Shokouhi conducted, especially when measured using MAP.
- However, even with this improved method of dividing result sets into segments, there were still elements of concern.
- In particular, adjacent documents could be treated 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

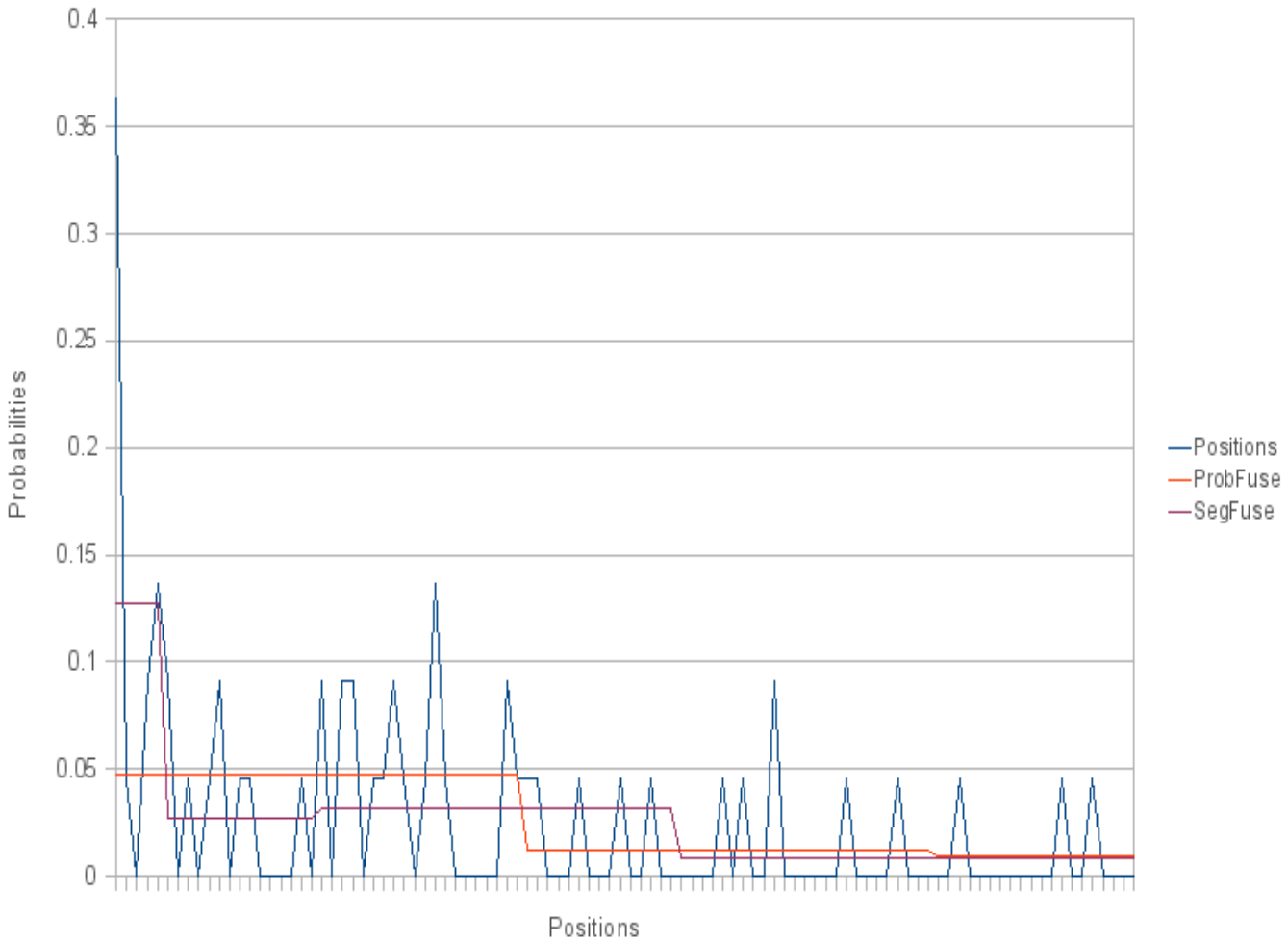
- This cutoff between segments means that documents that are side-by-side in a result set can be treated very differently.
- To deal with this, the *SlideFuse* algorithm was proposed.[3]
- Why not just use the probability at each rank in the result set?
- Especially for large-scale tasks, there may be 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.

[3] D. Lillis, F. Toolan, R. Collier and J. Dunnion, "Extending Probabilistic Data Fusion Using Sliding Windows", ECIR 2008

Distribution of Probabilities - Ranks



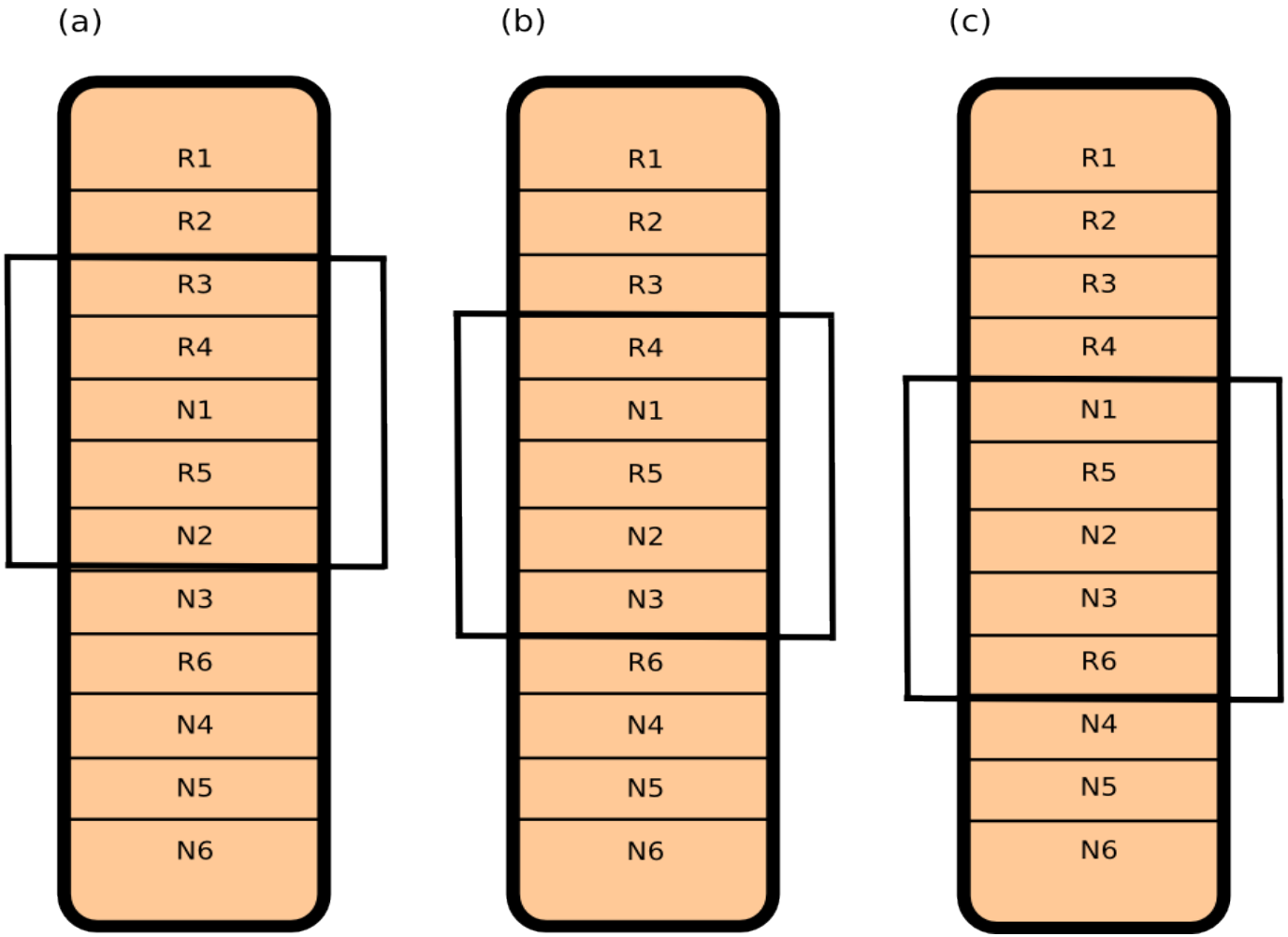
Distribution of Probabilities - ProbFuse and SegFuse



SlideFuse - Overview

- Instead of dividing the result sets into segments, we instead use each document's neighbours as evidence in estimating its probability.
- This will allow us to smooth the probability distribution graph so we avoid the jagged peaks seen when we use ranks only 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



SlideFuse - Training

- Training Phase - first calculate the probability of relevance at each rank
- $P(d_p|m) = \frac{\sum_{q \in Q_p} R_{d_p,q}}{Q_p}$
  - $P(d_p|m)$:= probability that document d returned in position p is relevant, given that it was returned by input model m
  - Q_p := set of training queries for which at least p documents were returned
  - $R_{d_p,q}$:= relevance of document d_p to query q (1 if relevant, 0 if not)

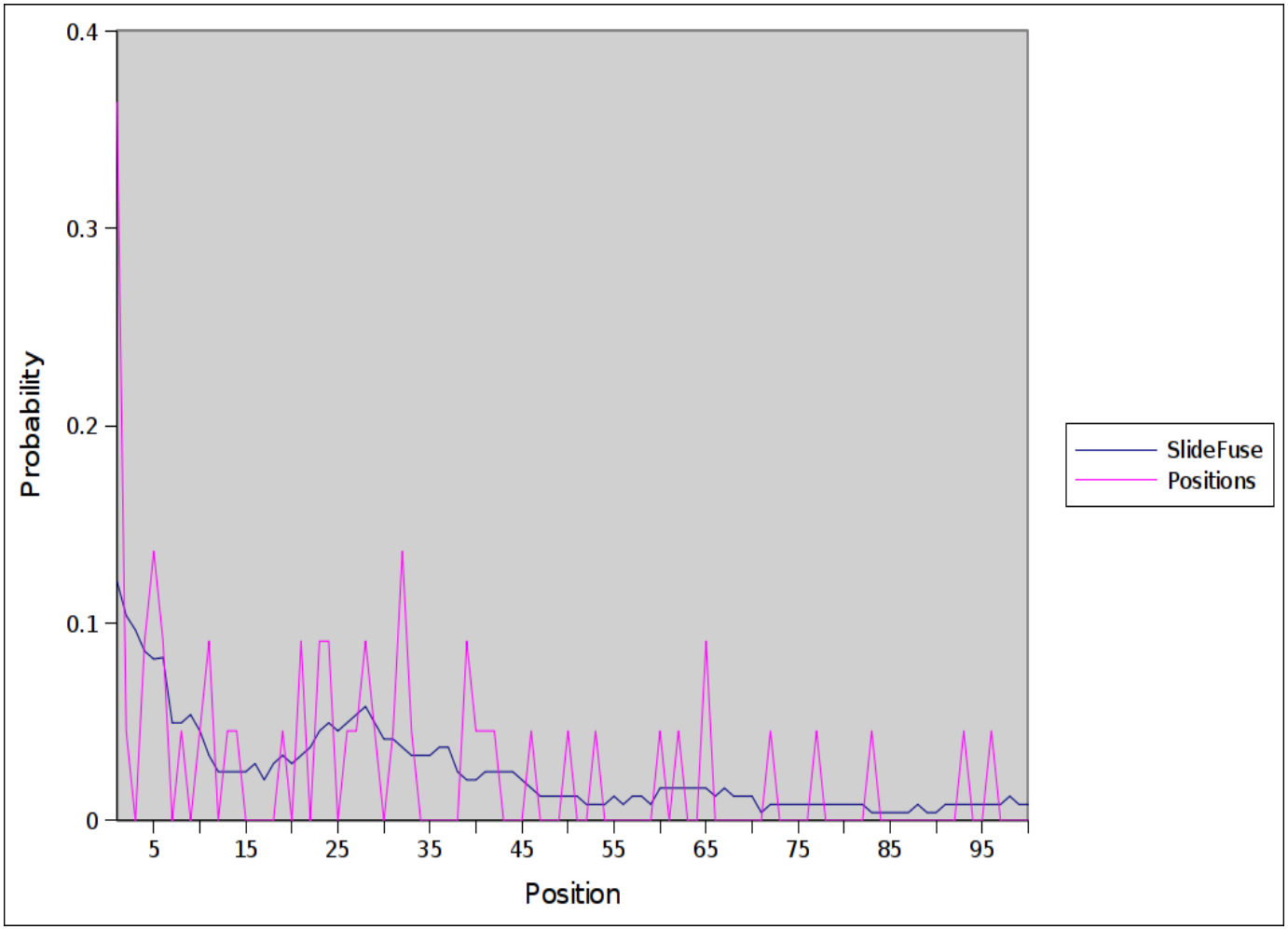
SlideFuse - Fusion

- Fusion Phase - assign probabilities to windows
- $P(d_{p,w}|m) = \frac{\sum_{i=a}^{b} P(d_i|m)}{b-a+1}$
  - $P(d_{p,w}|m)$:= probability that document d returned in position p is relevant, given that it was returned by input model m and the size of the window is w documents either side of p
  - a := position of first document in the window
  - b := position of last document in the window
  - $P(d_i|m)$:= probability that document d returned in position i is relevant, given that it was returned by input model m.

SlideFuse - Fusion

- Fusion Phase - calculate final ranking score
- $R_d = \sum_{m \in M} P(d_{p,w}|m)$
  - $R_d :=$ final ranking score for document d
  - $M :=$ set of all input models
  - $p :=$ position in which document d was returned

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 P10
- Average difference is $< 1\%$ in all other cases

Future Work

- Need for training data is limiting
- Currently assumes equal quality for each result set produced by an input system
- Weighted sliding windows, taking into account unjudged documents in the training phase
- Currently, documents returned in position beyond the length of the training result sets are not taken into account