Revisiting Optimal Rank Aggregation: A Dynamic Programming Approach

Shayan A. Tabrizi[1], Javid Dadashkarimi[1], Mostafa Dehghani[2], Hassan Nasr Esfahani[3], and Azadeh Shakery[1]

1: School of ECE, College of Engineering, University of Tehran, Iran
2: Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands
3: Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Problem

One of the most important applications of rank aggregation [3], that is merging multiple ranked lists of objects, is in the field of information retrieval (IR). Typically, rank aggregation in IR is conducted for two different purposes:

1. Aggregating ranked lists of a document set ranked by different ranking methods.
2. Aggregating ranked lists of different document sets ranked by the same ranking method. ▷ Our goal!

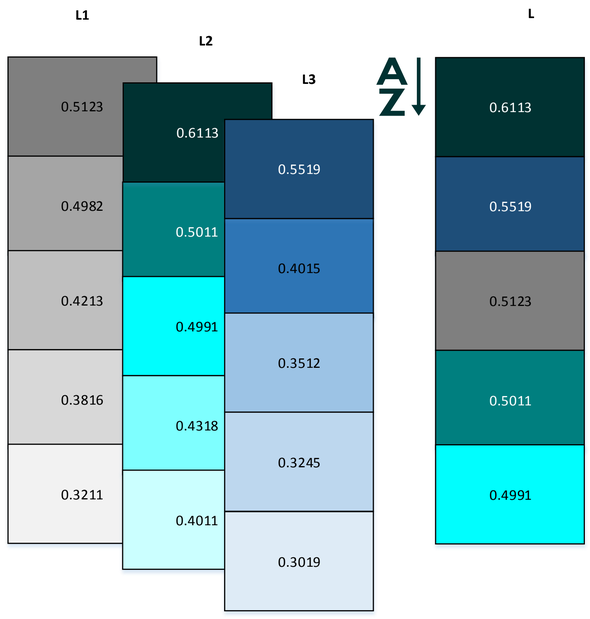

FIGURE 1: Rank aggregation problem: aggregating ranked lists of different document sets ranked by the same ranking method

Optimal Rank Aggregation: a rank aggregation problem in which, with the presence of relevancy information, items from different ranked lists are aggregated into a single final list so that:

- The final ranking has the best possible achievable performance;
- Local ordering of the initial lists are preserved in the final list.

Chen's Algorithm: Bucketing Strategy

A **bucket** is denoted in the form of $(n^-, n^+)\{d_i, d_{i+1}, \ldots, d_j\}$, in which n^- and n^+ respectively determine the number of **irrelevant** and **relevant** documents in the bucket and $\{d_i, d_{i+1}, \ldots, d_j\}$ denotes the documents ranked in positions i to j. For example $(3,1)\{d_3, d_4, d_5, d_6\}$ has 3 irrelevant documents, d_3, d_4, and d_5 and one relevant document d_6.

```
1: procedure ChenAggregator(Initial Ranked Lists)
2:     for each List l_i in the Initial Ranked Lists do
3:         b_i ← BucketRepresentation(l_i);
4:     end for
5:     MergedList ← ∅
6:     ActiveSet ← {b_i^1 | 1 ≤ i ≤ l ∧ n^+_{b_i^1} ≥ 1}
7:     repeat
8:         ActiveSet ← Sort(ActiveSet)
9:         b_k^j ← ActiveSet[1]
10:        MergedList.append(b_k^j)
11:        ActiveSet.remove(b_k^j)
12:        if n^+_{b_k^{j+1}} ≥ 1 then
13:            ActiveSet.add(b_k^{j+1})
14:        end if
15:    until isEmpty(ActiveSet)
16:    return MergedList
17: end procedure
```

FIGURE 2: Chen's algorithm [1, 2] for optimal rank aggregation

In this example, based on Chen's algorithm, the initial *active set* will be: $\{(0,1)\{A_1\}, (2,1)\{B_1, B_2, B_3\}, (1,3)\{C_1, C_2, C_3, C_4\}\}$. Continuing the algorithm, it will produce the following final result: $\{\{A_1\}, \{C_1, C_2, C_3, C_4\}, \{A_2, A_3\}, \{B_1, B_2, B_3\}, \{A_4\}, \{B_4\}\}$.

TABLE 1: The results of three different rankings. The relevant and irrelevant documents are marked by '+' and '−' respectively.

Rank	A	B	C
1	A_1^+	B_1^-	C_1^-
2	A_2^-	B_2^-	C_2^+
3	A_3^+	B_3^+	C_3^+
4	A_4^-	B_4^-	C_4^+

TABLE 2: Buckets of documents for the ranked lists.

Set	A	B	C
1	$(0,1)\{A_1\}$	$(2,1)\{B_1, B_2, B_3\}$	$(1,3)\{C_1, C_2, C_3, C_4\}$
2	$(1,1)\{A_2, A_3\}$	$(1,0)\{B_4\}$	
3	$(1,0)\{A_4\}$		

Counter-example

- The greedy approach does not necessarily lead to an optimal solution.
- In this example, the average precision of Chen's algorithm output is 0.5289, while that of the optimal aggregated list is 0.6198.

TABLE 3: The counterexample that shows that Chen's algorithm does not necessarily generate the optimal aggregated list.

Rank	A	B	Chen	DP
1	A_1^-	B_1^-	A_1^-	B_1^-
2	A_2^+	B_2^-	A_2^+	B_2^-
3	A_3^-	B_3^+	A_3^-	B_3^+
4	A_4^+	B_4^+	A_4^+	B_4^+
5	A_5^-	B_5^+	A_5^-	B_5^+
6	A_6^+	B_6^+	A_6^+	B_6^+
7	A_7^-	B_7^+	A_7^-	B_7^+
8	A_8^+	B_8^+	A_8^+	B_8^+
9			B_1^-	A_1^-
10			B_2^-	A_2^+
11			B_3^+	A_3^-
12			B_4^+	A_4^+
13			B_5^+	A_5^-
14			B_6^+	A_6^+
15			B_7^+	A_7^-
16			B_8^+	A_8^+
MAP			0.5289	0.6198

Proposed DP Algorithm

We introduce an m-th order tensor M of size $(|b_1|+1) \times (|b_2|+1) \times \ldots \times (|b_m|+1)$, in which value of an entry $M[i_1, i_2, \ldots, i_m]$ shows the maximum AP obtainable by aggregating top $i_1, i_2, \ldots, i_m$ buckets of the ranked lists $1, 2, \ldots, m$, respectively.

$$M[i_1, i_2, \ldots, i_m] = \max_k \big(M[i_1, i_2, \ldots, i_k - 1, \ldots, i_m] + \text{AP-GAIN}(k, i_1, i_2, \ldots, i_m)\big). \quad (1)$$

AP-GAIN$(k, i_1, i_2, \ldots, i_m)$ in Eq. 1 is the AP increase after appending $b_k^{i_k}$ at position p (i.e. at the end of the list):

$$p = \sum_{1 \le t \le m} \sum_{\substack{1 \le r \le i_t \wedge \\ \neg(t=k \wedge r=i_k)}} n_{b_t^r}. \quad (2)$$

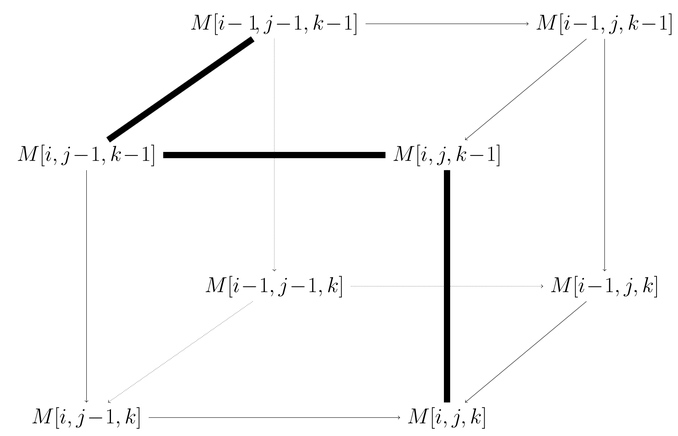

FIGURE 3: A graph of MAP belonging to bucket aggregation of 3 ranked lists. M[i,j,k] shows the maximum AP obtainable by aggregating top i,j, and k buckets the ranked lists 1,2 and 3 respectively.

```
1: procedure DPAgg(Bucket Representation)
2:     M ← [0]_{|b_1|×...×|b_m|}
3:     for i_1 ← 1 to |b_1| do
         ...
4:         for i_m ← 1 to |b_m| do
5:             v ← M[i_1−1, .., i_m] +AP-GAIN(1, i_1, .., i_m)
6:             k ← 1
7:             for k′ ← 2 to m do
8:                 ΔA_{k′} ←AP-GAIN(k′, i_1, ., i_{k′}−1, ., i_m)
9:                 v′ ← M[i_1, .., i_{k′}−1, .., i_m] +ΔA_{k′}
10:                if v′ ≥ v then
11:                    v ← v′
12:                    k ← k′
13:                end if
14:            end for
15:            M [i_1, ..., i_k − 1, ..., i_m] ← v
16:        end for
         ...
17:    end for
18:    return M
19: end procedure
```

FIGURE 4: DP algorithm for optimal rank aggregation

By storing index k maximizing Eq. 1 for each entry $M[i_1, i_2, \ldots, i_m]$, the optimal solution could be easily generated by starting from entry $M[|b_1|, |b_2|, \ldots, |b_m|]$ and iteratively inserting the bucket indicated by the stored index, at the beginning of the output.
Time complexity of:

- DP algorithm: $O(m \times |b_1| \times |b_2| \times \ldots \times |b_m| + n \times t)$.
- Brute-force algorithm: $O\left(\binom{|b_1|+|b_2|+\ldots+|b_m|}{|b_1|,|b_2|,\ldots,|b_m|}\right)$.

Acknowledgment

The authors would like to thank Google and SIGIR for providing a travel grant to attend this conference.

References

[1] A. Chen. Cross-language retrieval experiments at clef-2002. In *CLEF*, pages 28–48, 2002.

[2] A. Chen and F. C. Gey. Multilingual information retrieval using machine translation, relevance feedback and decom- pounding. *Information Retrieval*, 7(1-2):149–182, 2004.

[3] H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers, second edition, 2014.