Algorithm: Hybrid Exact Top-k with Threshold-Based k Selection

Input:

- Query q
- A set of items X
- Component-level embeddings $f_p(q)$, $g_p(x)$ for query q and item x
- Initial threshold T_{init} (a relevance score threshold)

Output:

- Top k items, G_{final}

Steps:

1. Initialize Candidate Set:

- Set $G \leftarrow \emptyset$ ▷ Initialize candidate set as empty

2. Generate Component-Level Embeddings:

For each $p \in P$, do:

- $X_p \leftarrow \{g_p(x) \mid x \in X\}$ ▷ Generate component-level embeddings for all items in X

3. Initial Candidate Retrieval:

For each $p \in P$, do:

- Compute dot product scores:

$$S_p = \{\langle f_p(q), g_p(x)\rangle : x \in X_p\}$$

- Retrieve items with scores $S_p \geq T_{\text{init}}$
- Add these items to G

4. Adjust k Dynamically:

- Compute MoL (Mixture of Logits) scores $s = \phi(q, x)$ for each $x \in G$ using:

$$\phi(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x)\rangle$$

  where $\phi(q, x)$ are weights that determine the importance of each component p.

- Set $T_{\text{adaptive}} = \min\{s : s \in G\}$ as the new threshold

5. Refine Candidate Set with Adaptive k:

For each $p \in P$, do:

- Use the updated threshold $T_adaptive$ to retrieve additional items from X
- Retrieve items from X_p with scores $S_p \geq T_{\text{adaptive}}$
- Add these items to G'

6. Select Exact Top-k Items:

- Compute MoL scores for all items in G'
- Sort G' by MoL scores in descending order
- Select the top k items from G' where k is the number of items in G' exceeding T_{adaptive}

7. Return Final Top-k Items:

$$G_{\text{final}} \leftarrow \text{Top } k \text{ items from } G'$$

Algorithm 1 Hybrid Exact Top-k with Threshold-Based k Selection

```
Input:
    • Query $q$
    • Set of items $X$
    • Component-level embeddings: $f_p(q)$, $g_p(x)$ for $p \in P$, $x \in X$
    • Initial threshold $T_{\text{init}}$
Output:
    • Exact top $k$ items based on dynamic threshold selection, $G_{\text{final}}$
1. Initialize:
    • Set $G \leftarrow \emptyset$                              ▷ Initial candidate set
2. Generate Component-Level Embeddings:
for each component $p \in P$ do
    $X_p \leftarrow \{g_p(x) \mid x \in X\}$                    ▷ Precompute embeddings
end for
3. Initial Candidate Retrieval:
for each component $p \in P$ do
    Compute dot product scores:
        $$S_p = \{\langle f_p(q), g_p(x) \rangle : x \in X_p\}$$
    Retrieve items with scores $S_p \geq T_{\text{init}}$
    Add these items to $G$
end for
4. Adjust k Dynamically:
for each $x \in G$ do
    Compute MoL scores $s = \phi(q, x)$ for each $x \in G$ using:
        $$\phi(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x) \rangle$$
    Set $T_{\text{adaptive}} = \min\{s : s \in G\}$
end for
5. Refine Candidate Set with Adaptive k:
for each component $p \in P$ do
    Retrieve items from $X_p$ with scores $S_p \geq T_{\text{adaptive}}$
    Add these items to $G'$
end for
6. Select Exact Top-k Items:
Compute MoL scores for all items in $G'$
Sort $G'$ by MoL scores in descending order
Select the top $k$ items from $G'$ where $k$ is the number of items in $G'$ exceeding
    $T_{\text{adaptive}}$
7. Return: $G_{\text{final}}$                                   ▷ Top $k$ items from $G'$
```