Multi-Task Learning for Email Search Ranking with Auxiliary Query Clustering

Jiaming Shen[1*], Maryam Karimzadehgan[2], **Michael Bendersky**[2], Zhen Qin[2], Donald Metzler[2]
[1]University of Illinois at Urbana-Champaign [2]Google Inc.
Presented by Michael Bendersky @ CIKM 2018

* Work done while interning at Google

How many people have searched the mailbox?

How many people will routinely clean the mailbox?

Email search comes to the rescue

Introduction

- Email search is widely used in people's daily life
- Email search queries have different types and come in very different flavors:
    - E.g., {“recent water bill”, “citi bank statement”} v.s. {“neural model papers”, “UMAI proposal”}

Simple reverse chronological ordering will be good enough

Relevance and content-based ranking is needed

- A single model fails to capture diverse ranking criterions

Goal: Exploit query-specific ranking models based on query type

Two Key Research Questions & Previous Studies

- **Research Question 1**: How to obtain query type information?
  - Previous approach 1: Train a query classifier using a labeled dataset
  - Previous approach 2: Cluster query using click data across different users
  - Limitation: both are inapplicable for email search due to the private nature of email
- **Research Question 2**: How to use query type information for ranking?
  - Previous approach: Train multiple type-specific ranking models (plus an optional global ranking model) and combine them together
  - Limitation: it’s burdensome to turn multiple ranking models in practice, and partitioning data causes each type-specific ranking model more likely to overfit

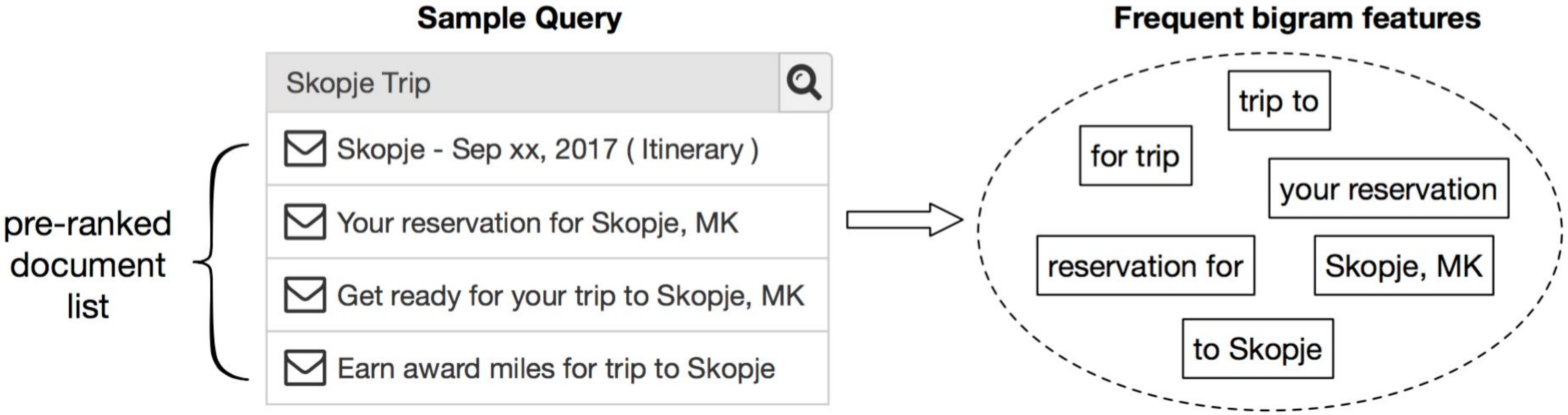
Two Key Research Questions & Previous Studies

- **Research Question 1**: How to obtain query type information?
  - Previous approach 1: Train a query classifier using a labeled dataset
  - Previous approach 2: Cluster query using click data across different users
  - Limitation: both are inapplicable for email search due to the private nature of email
  - Our solution: Hierarchical query clustering with document-enhanced query representation
- **Research Question 2**: How to use query type information for ranking?
  - Previous approach: Train multiple type-specific ranking models (plus an optional global ranking model) and combine them together
  - Limitation: it’s burdensome to turn multiple ranking models in practice, and partitioning data causes each type-specific ranking model more likely to overfit
  - Our solution: Joint ranking and query cluster prediction within a multi-task learning framework

Our Method

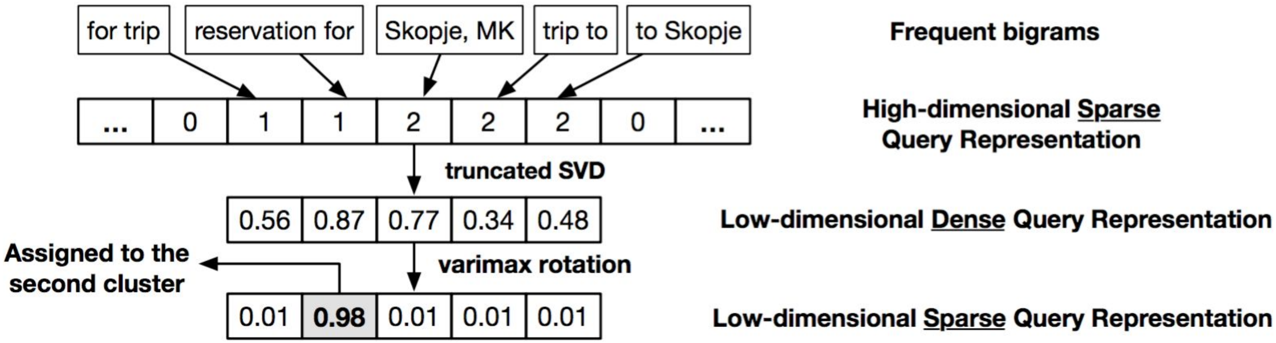
Hierarchical Query Clustering -- Query Representation

- Key challenge: limited query features due to privacy
- Our method:
  - First, leverage a reasonable base ranker (e.g., BM25) to obtain a pre-ranked list
  - Second, extract features from top documents in the pre-ranked list
  - Finally, combine document-level features with query original features



Hierarchical Query Clustering -- Query Clustering

- Key challenge: work on high-dimensional features and need to scale
- Our method:
  - Step 1: use **truncated SVD model** (a.k.a LSI) to convert *sparse, high-dimensional* feature vectors into *dense, low-dimensional* feature vectors
  - Step 2: use **varimax rotation** to project *dense, low-dimensional* feature vectors into a few of axes and obtain *sparse, low-dimensional* feature vectors
  - Finally, **recursively apply step 1&2 in a top-down fashion**



Hierarchical Query Clustering -- Query Clustering

Algorithm 1: Divisive Hierarchical Query Clustering

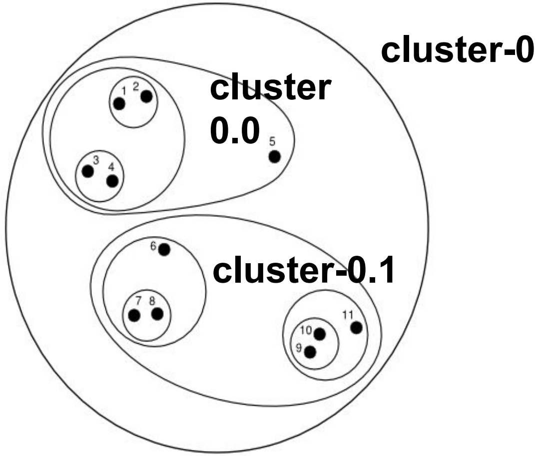
Input: A collection of queries $\mathbb{Q}$; the depth of hierarchy D; the number of branch in each level B; the minimal required examples in each leaf node E.

Output: A hierarchical cluster tree T of queries $\mathbb{Q}$.

Assign all queries to root cluster at depth $d = 0$;
Initialize $T \leftarrow \emptyset$;
for *depth d from 0 to D* **do**
    for *cluster c at depth d* **do**
        $S_c \leftarrow$ queries assigned to c;
        if $d = D$ **then**
            if $|S_c| < E$ **then**
                $T \leftarrow T - \{c\}$ // Prune this leaf node;
            Continue;
        $F_c \leftarrow \text{VARIMAX}(\text{TRUNCATED-SVD}(S_c, B))$;
        $N_1, \ldots, N_B \leftarrow \text{QUERY-ASSIGN}(F_c)$;
        for *sub cluster index i from 1 to B* **do**
            $T \leftarrow T \cup N_i$;
Return hierarchical cluster tree T;

Initially, all queries are in root cluster

Recursively divide queries in current cluster into multiple sub-clusters



Hierarchical Query Clustering -- Summary

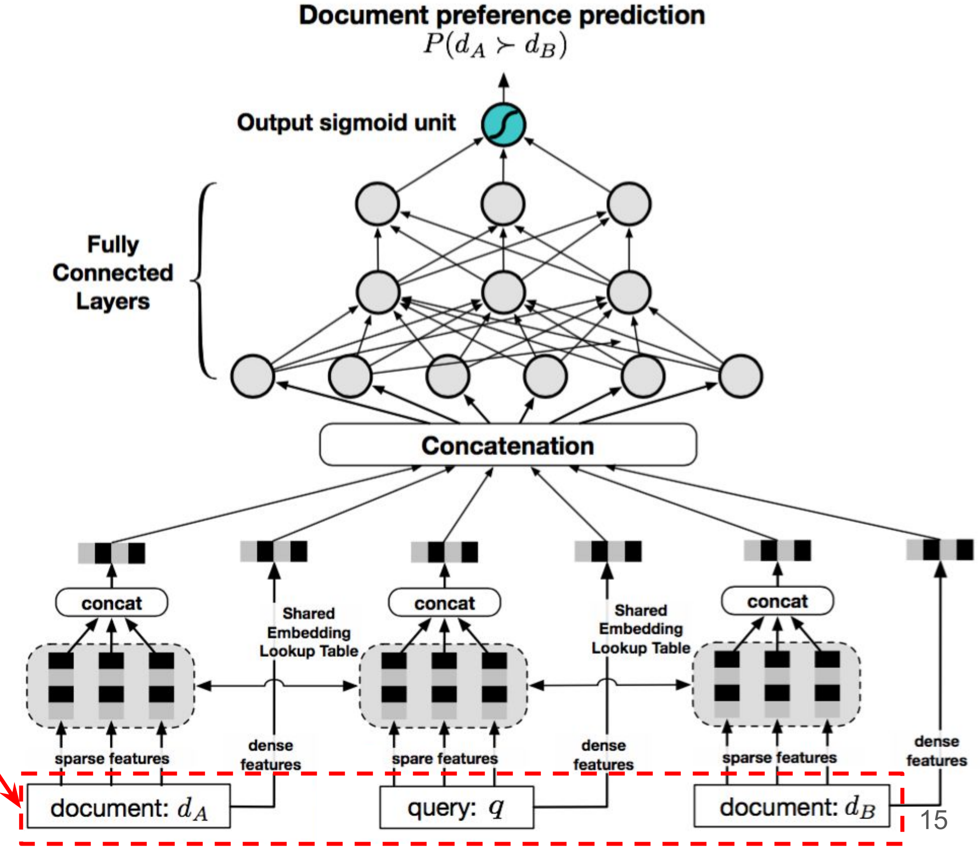
- We map each query into multiple clusters of different granularities
  - E.g. query-1 -> [“cluster-0”, “cluster-1”, “cluster-1.3”, “cluster-1.3.4”]

Root cluster 1st level cluster 2nd level cluster 3rd level cluster

- Advantage of our method:
  - Able to leverage sparsity in original high-dimensional feature vectors
  - Scale to billions of examples
  - Provide clusters of different granularities and allow subsequent models to pick the adequate level of granularity for each cluster

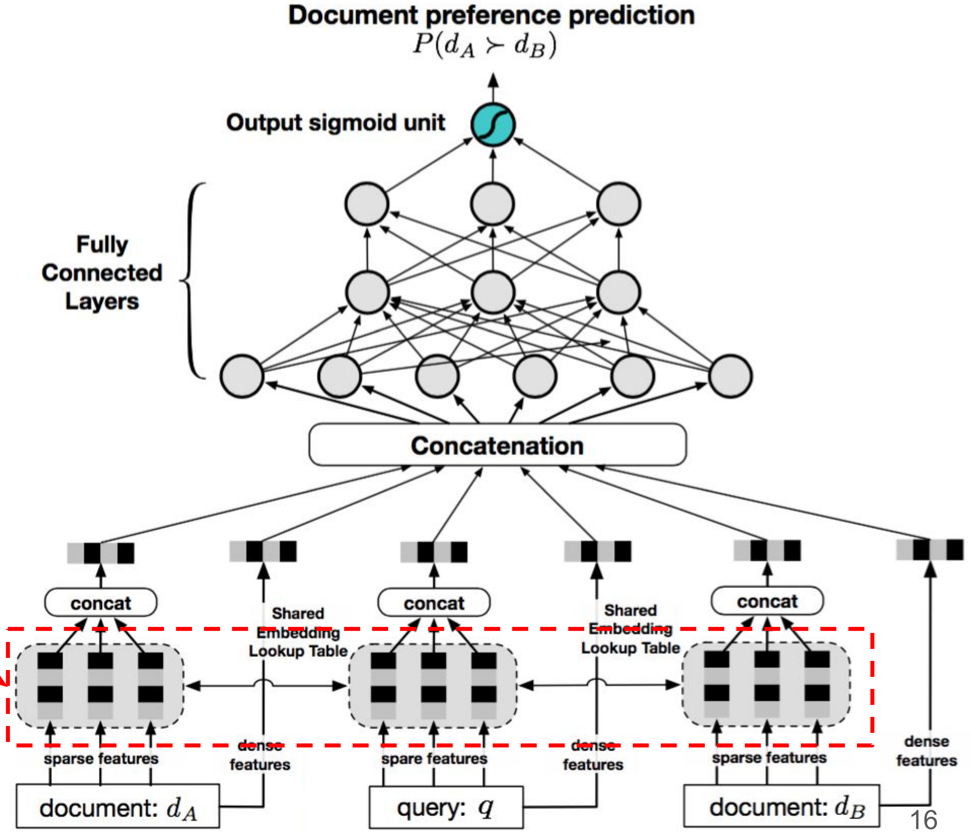
Background: Query-independent Ranking

- Use pairwise ranking paradigm:
  - A query
  - A positive document
  - A negative document



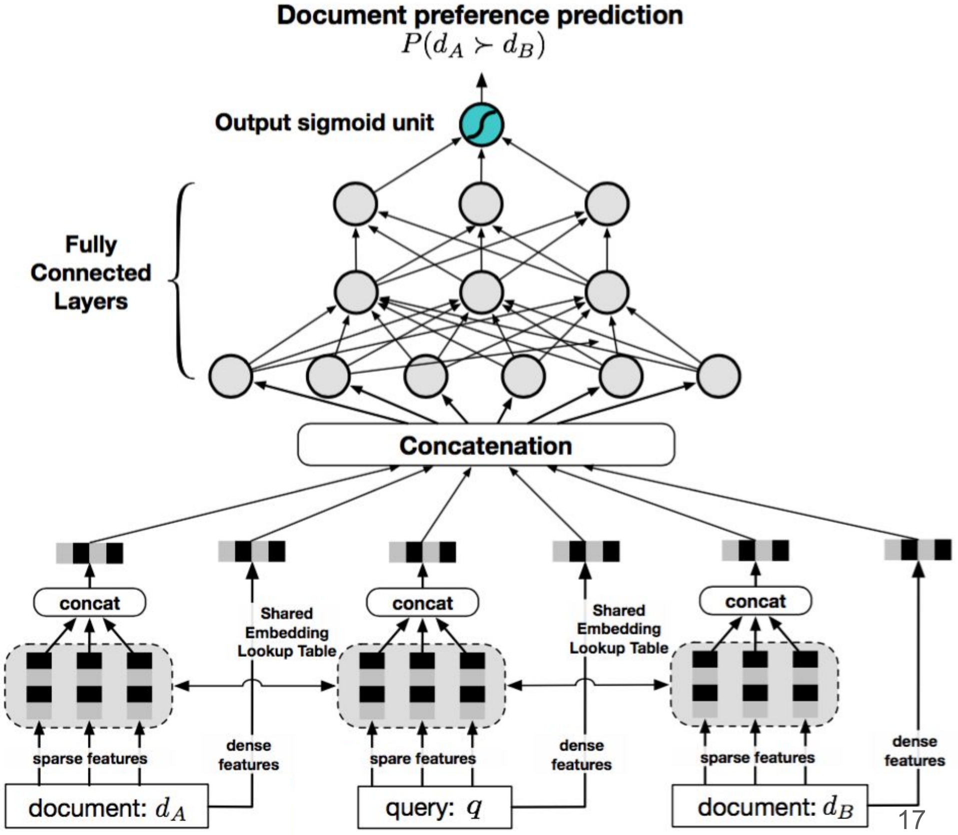
Background: Query-independent Ranking

- Use pairwise ranking paradigm:
  - A query
  - A positive document
  - A negative document
- Use embedding for sparse features



Background: Query-independent Ranking

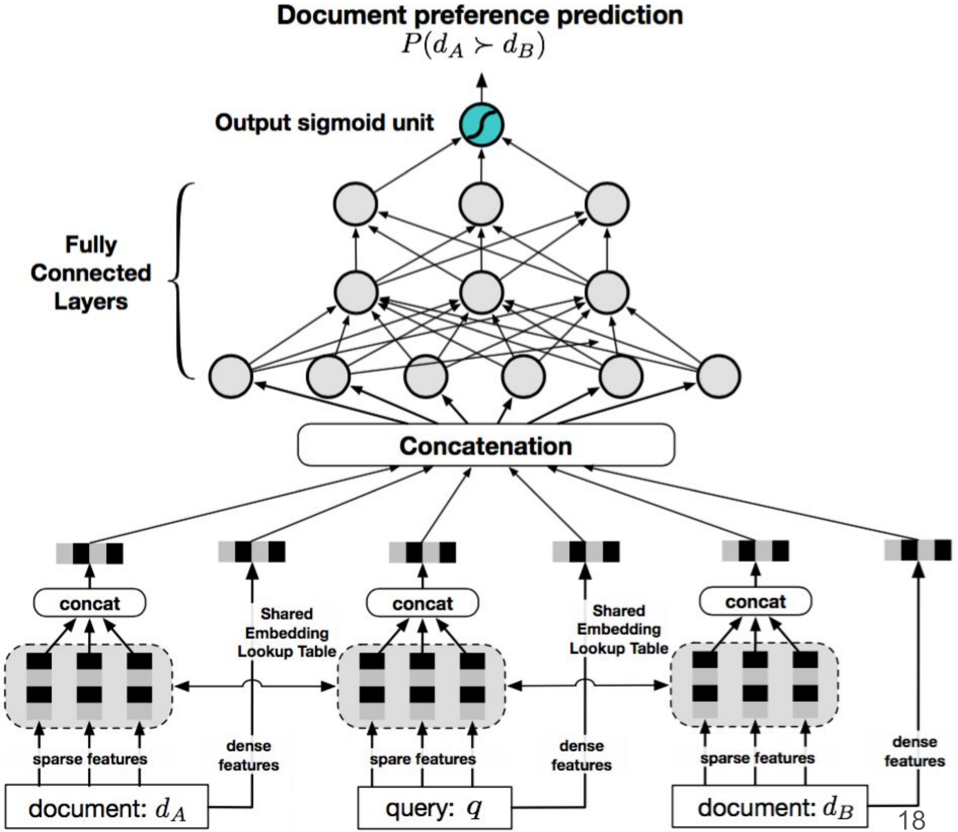
- Use pairwise ranking paradigm:
  - A query
  - A positive document
  - A negative document
- Use embedding for sparse features



Deep Pairwise Ranking Model (DPRM)

Background: Query-independent Ranking

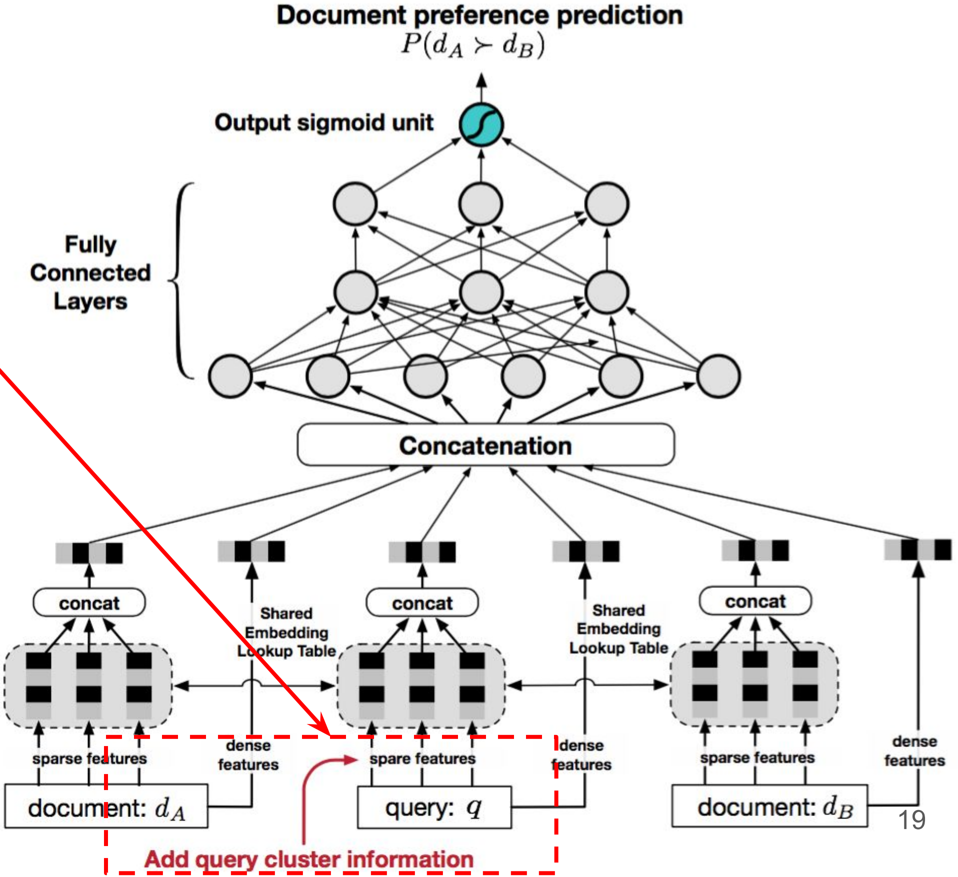
- Use pairwise ranking paradigm:
  - A query
  - A positive document
  - A negative document
- Use embedding for sparse features
- Our goals:
  - Incorporate query type information
  - Achieve query-dependent ranking



Deep Pairwise Ranking Model (DPRM)

First Attempt: QC-DPRM

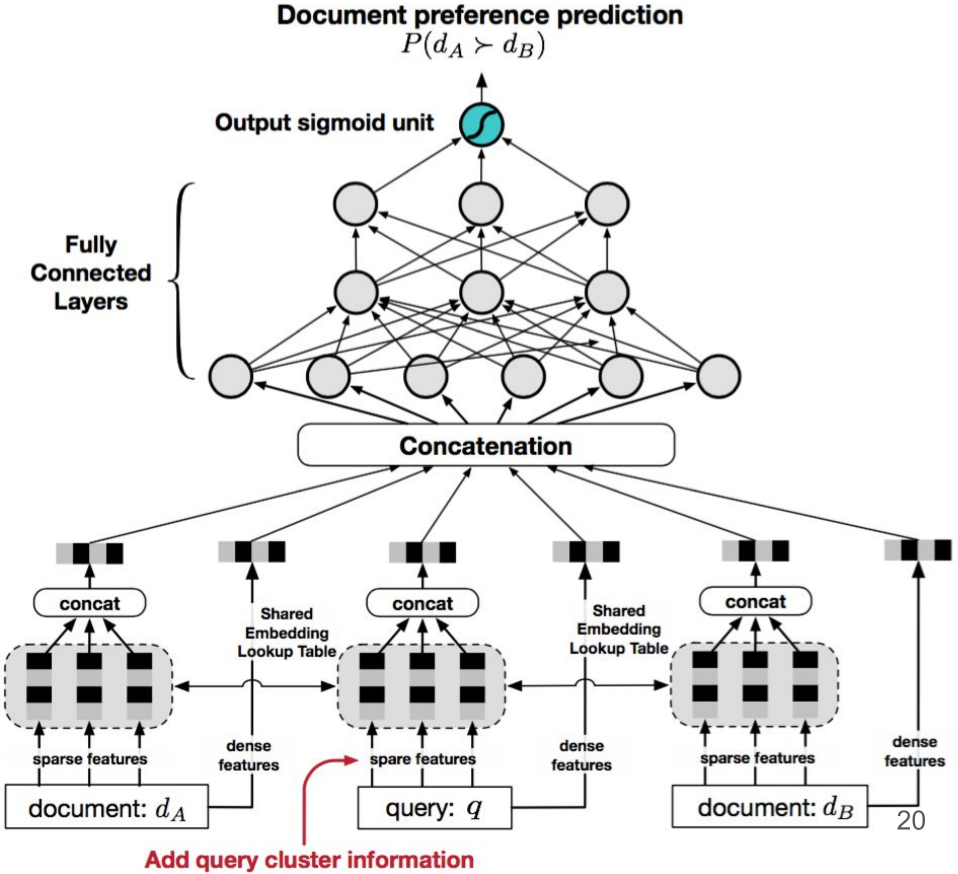
- Add query clusters as features



Query-Cluster aware Deep Pairwise Ranking Model (QC-DPRM)

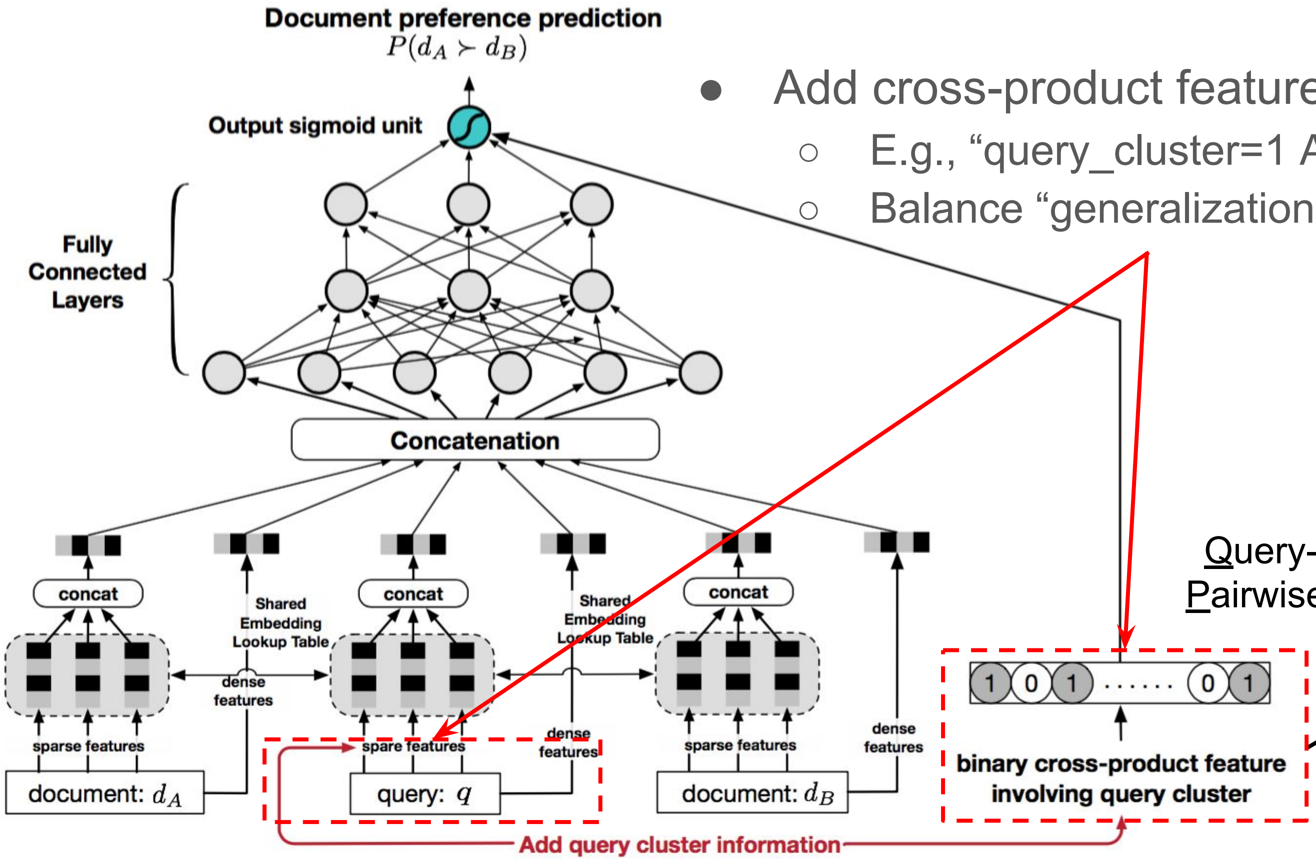
First Attempt: QC-DPRM

- Add query clusters as features
- Limitations:
  - Cannot distinguish cluster features with other query sparse features



Query-Cluster aware Deep Pairwise Ranking Model (QC-DPRM)

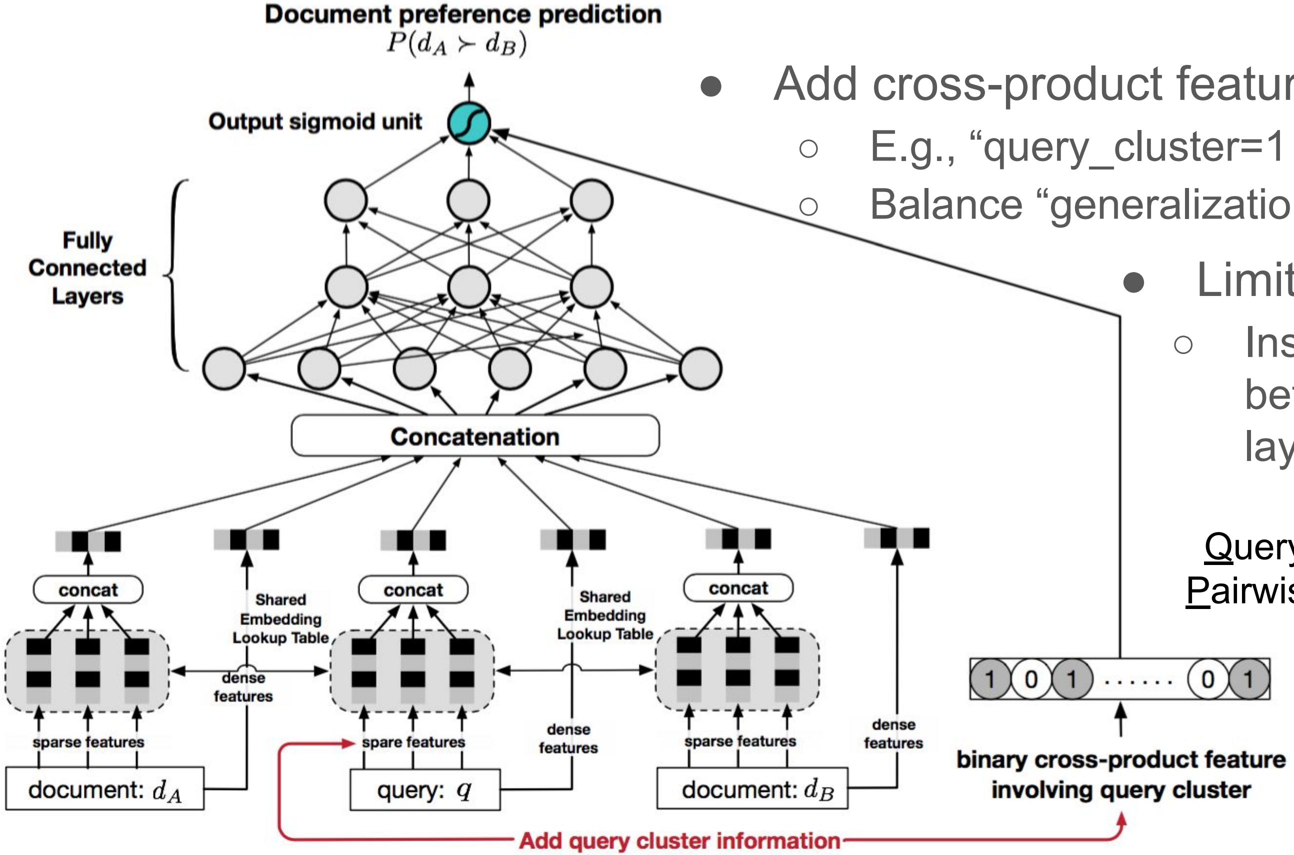
Second Attempt: QC-WDPRM



- Add cross-product features in a "wide" linear model
  - E.g., "query_cluster=1 AND language=English"
  - Balance "generalization" and "memorization"

Query-Cluster aware Wide and Deep Pairwise Ranking Model (QC-WDPRM)

Second Attempt: QC-WDPRM



- Add cross-product features in a "wide" linear model
  - E.g., "query_cluster=1 AND language=English"
  - Balance "generalization" and "memorization"

- Limitations:
  - Insufficient to model interactions between features from input layers

Query-Cluster aware Wide and Deep Pairwise Ranking Model (QC-WDPRM)

Query-dependent Ranking by Multi-Task Learning

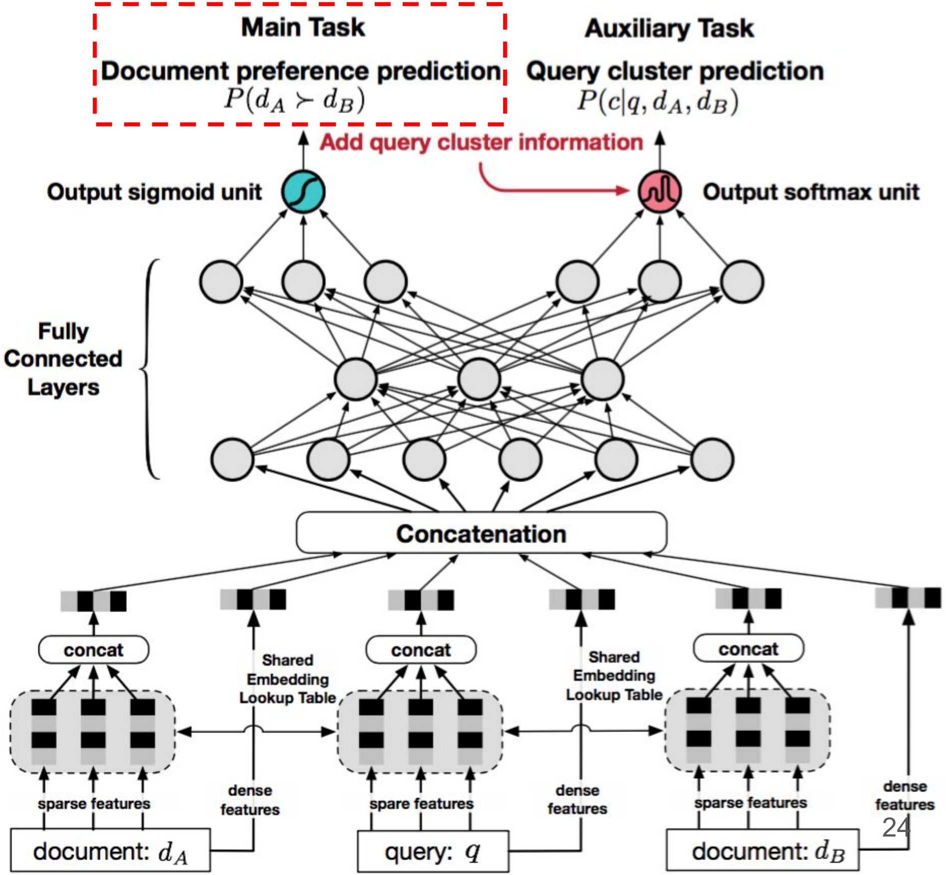
- Motivations:
  - Back to the initial thinking, we aim to share the training of multiple ranking models, one for each query type
  - Push the query cluster feature in a top-down fashion to influence of all query/document representation learning
- Core idea: use multi-task learning to combine two tasks
  - Main Task: Email Search Ranking
  - Auxiliary Task: Query Cluster Prediction
- We present our Query-Cluster aware Multi-Task Learning Ranking Model (QC-MTLRM)

Query-dependent Ranking -- MLTRM

- Main Task -- ranking loss:

Equals to 1 if *document a* is preferred to *document b* and equals to 0 if otherwise

$$
\begin{aligned}
l^{rank}(q) &= -\boxed{y_{ab}}\log(p_{ab}) - (1-\boxed{y_{ab}})\log(1-p_{ab}),\\
p_{ab} &= P(d_A \succ d_B),
\end{aligned}
$$



Query-dependent Ranking -- MLTRM

- Main Task -- ranking loss:

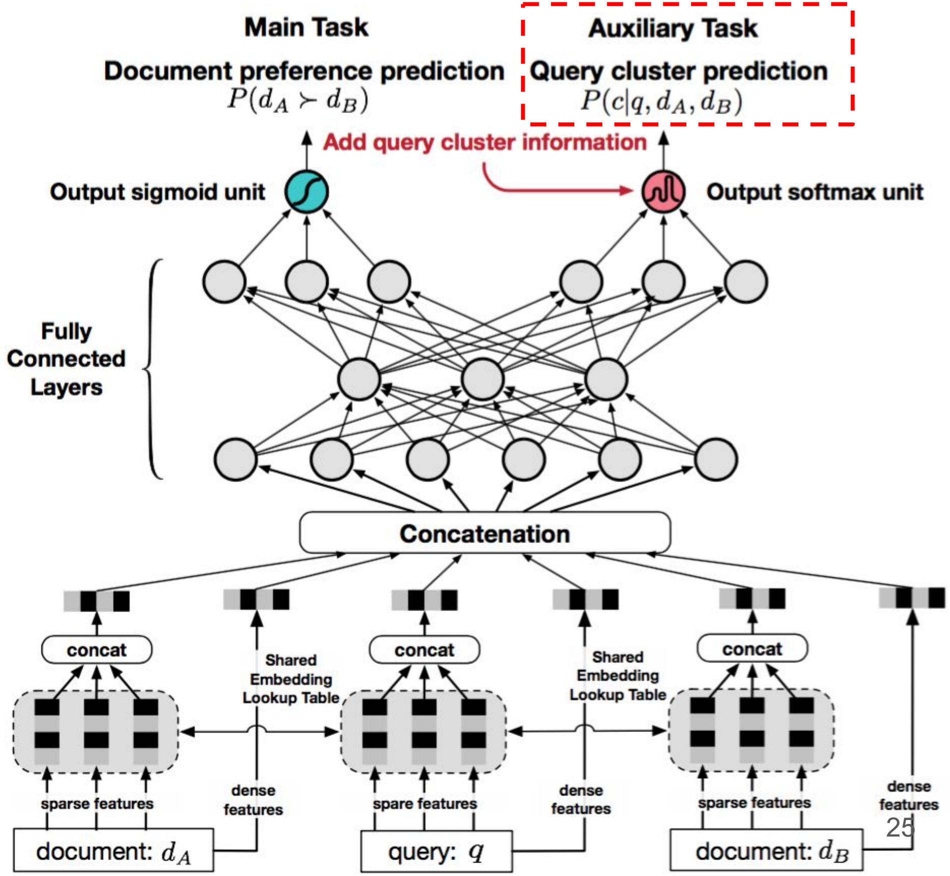
Equals to 1 if *document a* is preferred to *document b* and equals to 0 if otherwise

$$
\begin{aligned}
l^{rank}(q) &= -\boxed{y_{ab}}\log(p_{ab}) - (1-\boxed{y_{ab}})\log(1-p_{ab}),\\
p_{ab} &= P(d_A \succ d_B),
\end{aligned}
$$

- Auxiliary Task -- classification loss:

$$
\begin{aligned}
l^{cluster}(q) &= -\sum_{c\in C} \boxed{p_c} \log \hat{p_c},\\
\hat{p_c} &= p(c|q, d_A, d_B),
\end{aligned}
$$

Ground truth query cluster



Query-dependent Ranking -- MLTRM

- Main Task -- ranking loss:

Equals to 1 if *document a* is preferred to *document b* and equals to 0 if otherwise

$$l^{rank}(q) = -\boxed{y_{ab}}\log(p_{ab}) - (1-\boxed{y_{ab}})\log(1-p_{ab}),$$
$$p_{ab} = P(d_A \succ d_B),$$

- Auxiliary Task -- classification loss:

$$l^{cluster}(q) = -\sum_{c\in C}\boxed{p_c}\log\hat{p_c},$$

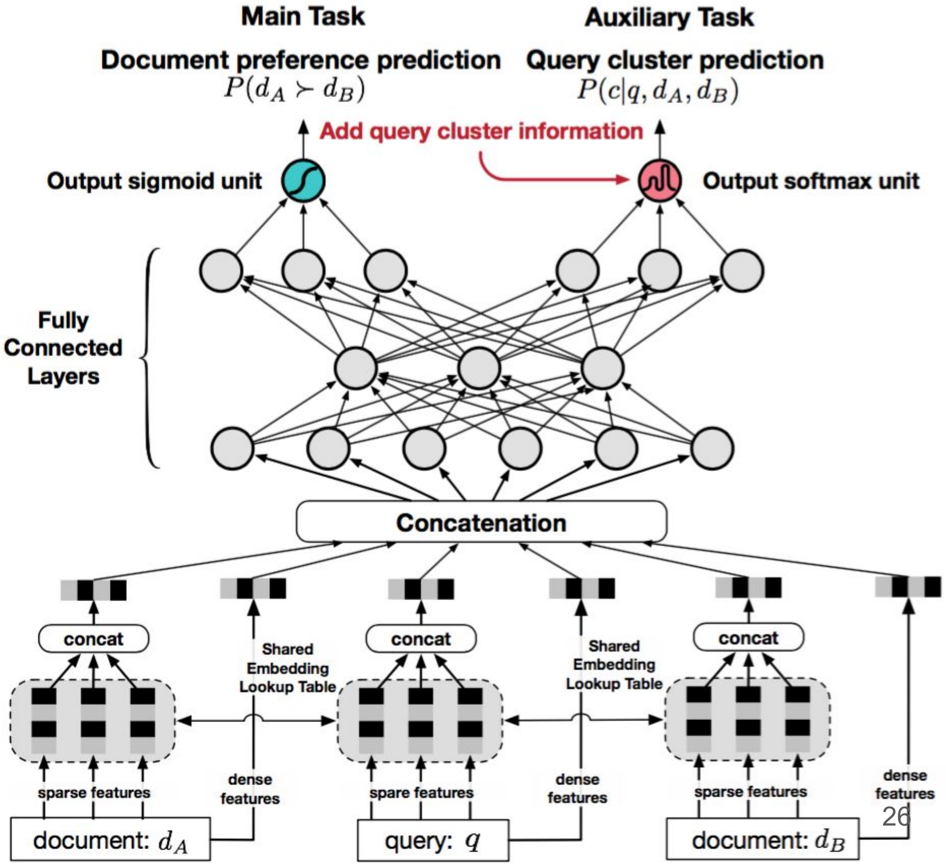
Ground truth query cluster

$$\hat{p_c} = p(c|q, d_A, d_B),$$

- Combined loss:

mix_rate

$$L(\Theta) = \frac{1}{|Q|}\sum_{q\in Q}\left(l^{rank}(q) + \boxed{\lambda}\, l^{cluster}(q)\right)$$



Experiments

Experimental Setup

- Evaluation dataset:
  - Anonymized Gmail queries with *k*-anonymity approach
  - 66 million training queries, 4 million validation queries, and 9 millions testing queries, splitted based on their issued time to avoid data leakage
- Query and document features:

Feature Type	**Descriptions**	**Usage**
Content	List of frequent n-grams appearing in the query text and the email subject *e.g., "Class schedule on Friday morning" → ["class schedule", "Friday morning"].*	Cluster queries Learn ranking models
Category	Small set of commonly used email labels *e.g., Promotions, Forums, Purchases, and etc.* (see [2] for detailed label examples)	Cluster queries Learn ranking models
Structure	Frequent machine-generated email subject templates *e.g., Your trip confirmation number 12345 → Your trip confirmation number* * (see [2] Table 2 for more details on structure features)	Cluster queries Learn ranking models
Situational	Temporal and Geographical features of current search request *e.g., Friday, 8:00pm, USA, Japan* (see [42] for more details)	Learn ranking models

Experimental Setup

- Evaluation metrics:
  - Mean Reciprocal Rank (MRR)
  - success@1: percentage of queries for which clicked email is ranked in top-1
  - success@5: percentage of queries for which clicked email is ranked within top-5
- Hyper-parameters:
  - Depth and number of branches in the hierarchical clustering algorithm
  - Number of hidden layer, hidden layer size, learning rate, drop-out, embedding size, optimization algorithm
  - Tuned on validation set

Q1: How do different query-dependent ranking models leverage the query cluster information?

- Treating query cluster information as additional feature does not work well
- Incorporating query cluster information as “label” for auxiliary query cluster prediction task really helps

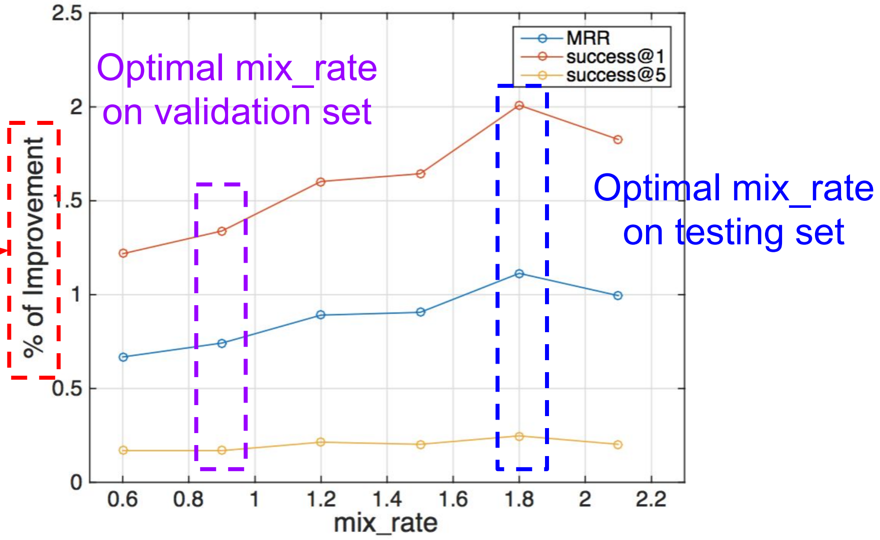
Method	**MRR**	**success@1**	**success@5**
DPRM	0.6698	0.4874	0.8861
QC-DPRM	0.6697 (-0.01%)	0.4873 (-0.02%)	0.8864 (+0.03%)
QC-WDPRM	0.6699 (+0.01%)	0.4875 (+0.02%)	0.8862 (+0.01%)
QC-MTLRM	**0.6748 (+0.70%)***	**0.4939 (+1.32%)***	**0.8875 (+0.17%)***

* indicates the improvement is statistically significant in terms of two-tailed paired t-test with 99% confidence level

Q2: How do different mix_rates influence the performance of query-dependent ranking models?

- We use mix_rate to balance the ranking loss with cluster prediction loss
- Applying multi-task learning with a wide range of mix_rate can help improve the ranking performance

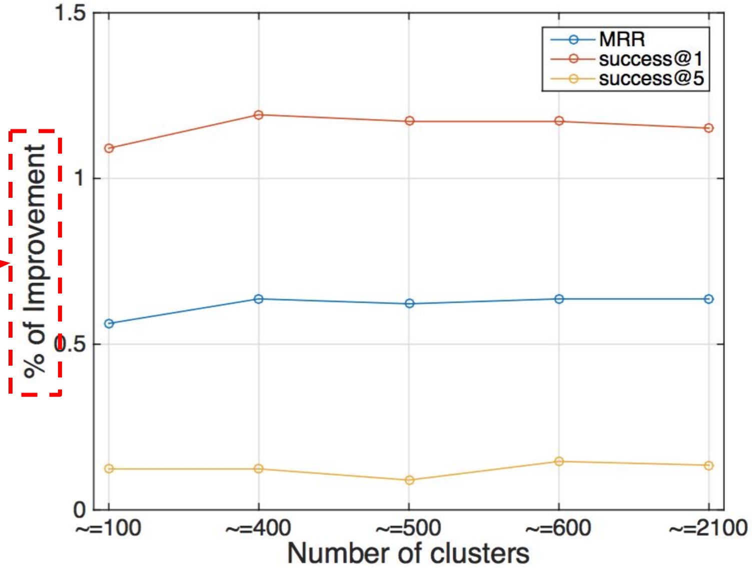
Relative improvement of QC-MTLRM over DPRM



Q3: How do different cluster numbers influence the performance of query-dependent ranking models?

- We find QC-MTLRM is insensitive to the cluster number
  - We suspect the reason is that around 100 query clusters can capture most of the important data-dependent information

Relative improvement of QC-MTLRM over DPRM



Q4: How do different query-dependent ranking models contribute to an end-to-end ranking pipeline

- Production-level search engines usually have an end-to-end ranking pipeline which integrates multiple ranking signals
- Weighted Average Click Position (WACP): smaller the better
- QC-MTLRM can effectively leverage query cluster information and output indicative signals useful to an end-to-end ranking pipeline

Method	**WMRR**	**WACP**
LTR + DPRM	+2.35%*	-3.24%*
LTR + QC-DPRM	+2.32%*	-3.20%*
LTR + QC-WDPRM	+2.35%*	-3.28%*
LTR + QC-MTLRM	**+2.52%****	**-3.41%****

* indicates the improvement is statistically significant over LTR and ** indicates the improvement is statistically significant over both LTR and

Conclusions & Future Work

- **Goal:**
  - Exploit query-specific ranking models for different (types of) queries
- **Methods**:
  - Use hierarchical query cluster to obtain query type information
  - Use multi-task learning to leverage query type information in ranking model
- **Future Directions:**
  - Leverage hierarchical query clustering algorithm to obtain user clusters and then build user-specific ranking model
  - Extend the multi-task learning idea to pointwise/listwise ranking paradigms

Thank you!

Questions?