Leveraging User Behavior History for Personalized Email Search

Keping Bi[1], Pavel Metrikov[2,4], Chunyuan Li[3], Byungki Byun[2]

[1] University of Massachusetts Amherst, [2] Microsoft, [3] Microsoft Research, USA

[4] Institute for Information Transmission Problems, Russian Academy of Sciences, Russia

kbi@cs.umass.edu, {pametrik, chunyl, bybyun}@microsoft.com

ABSTRACT

An effective email search engine can facilitate users' search tasks and improve their communication efficiency. Users could have varied preferences on various ranking signals of an email, such as relevance and recency based on their tasks at hand and even their jobs. Thus a uniform matching pattern is not optimal for all users. Instead, an effective email ranker should conduct personalized ranking by taking users' characteristics into account. Existing studies have explored user characteristics from various angles to make email search results personalized. However, little attention has been given to users' search history for characterizing users. Although users' historical behaviors have been shown to be beneficial as context in Web search, their effect in email search has not been studied and remains unknown. Given these observations, we propose to leverage user search history as query context to characterize users and build a context-aware ranking model for email search. In contrast to previous context-dependent ranking techniques that are based on raw texts, we use ranking features in the search history. This frees us from potential privacy leakage while giving a better generalization power to unseen users. Accordingly, we propose a *context-dependent neural ranking model* (CNRM) that encodes the ranking features in users' search history as query context and show that it can significantly outperform the baseline neural model without using the context. We also investigate the benefit of the query context vectors obtained from CNRM on the state-of-the-art learning-to-rank model LambdaMart by clustering the vectors and incorporating the cluster information. Experimental results show that significantly better results can be achieved on LambdaMart as well, indicating that the query clusters can characterize different users and effectively turn the ranking model personalized.

CCS CONCEPTS

• **Information systems** → **Environment-specific retrieval**; **Retrieval models and ranking**.

KEYWORDS

Email Search, Query Context, Personalized Search

1 INTRODUCTION

Email has long been an indispensable way for both business and personal communication in our daily life. As email storage quotas become large due to cheap data storage, users do not feel the need to delete emails and let their mailboxes keep growing [9, 20]. Therefore, an effective email search engine is quite important to support users' search needs and facilitate users' daily life.

In email search, users could have varied preferences on email's various ranking signals such as relevance, recency, and other email properties due to their primary purposes of the accounts and their occupation. For instance, commercial users (the employees of a company that uses the email service) and consumer users (individual email service users) could have significantly different preferred email properties, such as the length of the email threads and the number of recipients and attachments. While it can be known explicitly whether a user is commercial or consumer, other implicit user cohorts also exist for whom different aspects of emails matter more for ranking. For example, in contrast to engineers, it is more likely that the emails salespeople want to re-find have them as senders rather than as recipients. Users who organize their mailboxes more often could prefer recency than relevance since they can search inside a specific folder that already satisfies some filtering conditions. Hence, it could improve search quality by identifying diversified implicit user cohorts as query context and conducting personalized ranking accordingly.

Existing work has explored various types of user information to provide personalized email search results. Zamani et al. [38] leverage the situational context of a query such as geographical (e.g., country, language) and temporal (e.g., weekday, hour) features to characterize users. Weerkamp et al. [34] used users' email threads and mailing lists as contextual information to expand queries. Kuzi et al. [15] trained local word embeddings with the content of each user's mailbox and used the nearest neighbors of the query terms in the embedding space as the query context. Shen et al. [27] collected the frequent n-grams of the results retrieved from the user's mailbox with a decent initial ranker, based on which the query is clustered, and the cluster information can be considered as context for conducting adaptive ranking.

Despite the extensive studies on exploring user characteristics for personalized email search, little attention has been paid to characterizing users with their search history. Leveraging users' historical click-through data as context has been shown to be beneficial in Web search [6, 13, 17, 21, 28, 31, 35, 36]. However, the effect of incorporating users' historical behaviors in email search has not been studied and remains unknown. There are substantial differences between email search and Web search. To name a few, first, email search can only be conducted on users' personal corpus instead of a shared public corpus, and the target is usually to re-find a known item; second, email content cannot be exposed to third parties due

to privacy concern, which makes it hard to investigate and analyze ranking models; third, in email search logs, cross interactions with the same item from different users do not exist and search queries may be hard to generalize across users since they are personal and pertain to their particular email content. Due to the differences mentioned above, it is valuable to investigate the benefit of using user search history as context for email search, and it is also challenging to incorporate this information effectively.

In this paper, we study the effect of characterizing users with their search history as query context in personalized email search. As far as we know, this is the first work in this direction. We construct the context from user history grounded on ranking features instead of raw texts as previous context-aware ranking models do. These numerical features free us from potential privacy leakage present in a neural model trained with raw texts while having better generalization capabilities to unseen users. They can also capture users' characteristics from a different perspective compared with their semantic intents indicated by the terms or n-grams in their click-through data. Based on both neural models and the prevailing state-of-the-art learning-to-rank model - LambdaMart [8], we investigate the query context obtained from the ranking features in the user search history. Specifically, we propose a neural baseline ranking model based on the numerical ranking features and a context-dependent neural ranking model (CNRM) that encodes the ranking features of users' historical queries and their associated clicked documents as query context. We further cluster each query context vector learned from CNRM to reveal hidden user cohorts that are captured in users' search behaviors. To examine the effectiveness of using the learned user cohorts, we then incorporate the cluster information into the LambdaMart model. Experimental results show that the query context helps improve the ranking performance on both neural models and LambdaMart models.

Our contributions in this paper can be summarized as follows: 1) we propose a context-dependent neural ranking model (CNRM) to incorporate the query context encoded from numerical features extracted from users' search history to characterize potential different matching patterns for diversified user groups; 2) we conduct a thorough evaluation of CNRM on the data constructed from the search logs of one of the world's largest email search engines and show that CNRM can outperform the baseline neural model without context significantly; 3) we cluster the query context vectors and incorporate the query cluster information with LambdaMart and produce significantly better ranking performance; 4) we analyze the information carried in the query context vectors, the feature weight distribution of the query clusters in the LambdaMart model, and users' distribution in terms of their query clusters.

2 RELATED WORK

Our paper is directly related to two threads of work: email search and context-aware ranking in web search.

Email Search. There have been limited existing studies on email search, probably due to the lack of publicly available email data, which is too private and sensitive to expose. The TREC Enterprise track dataset [11, 29] is an exception. The dataset contains about 200k email messages crawled from the W3C public mailing list

archive [1] and 150 <query, document-id> pairs for email re-finding tasks. Earlier studies on email search are mainly based on this dataset. Macdonald and Ounis [19] proposed a field-based weighting model that can combine Web features with email features and investigated where field evidence should be used or not. Craswell et al. [12] scored email messages based on their multi-field with the BM25f formula [26]. Ogilvie and Callan [23] combined evidence from the subject, content of an email, the thread the email occurs in, and the content in the emails that are in reply to the email with a language model approach. Yahyaei and Monz [37] conducted maximum entropy modeling to estimate feature weights in known-item email retrieval. Weerkamp et al. [34] studied to improve email search quality by expanding queries with information sources such as threads and mailing lists.

Some research also uses two other datasets to study email search which are Enron [2] and PALIN [3] [1, 7]. Enron is an enterprise email corpus that the Federal Energy Regulatory Commission released after the Enron investigation. PALIN is the personal email corpus of the former Alaskan governor Sarah Palin. Both of them do not have queries or relevance judgments of emails. AbdelRahman et al. [1] generated 300 synthetic queries for Enron using topic words related to Enron and estimate email relevance by three judges according to given guidelines. Bhole and Udupa [7] conducted misspelling query correction on queries constructed by volunteers for both Enron and PALIN. Similar to the W3C TREC dataset, they are far from real scenarios as the queries, and the relevance judgments do not originate from the mailbox owners.

In recent years, more studies have been conducted with real queries and judgments derived from users' interactions with the results in the context of large Web email services. Some work has studied the effect of displaying results ranked based on relevance or recency [9, 10, 25]. Carmel et al. [9] followed a two-phase ranking scheme where in the first stage recall is emphasized based on exact or partial term matching, and in the second stage, a learning-to-rank (LTR) model is used to balance the relevance and recency for ranking. This way outperforms the widely adopted time-based ranking by a large margin. Aware that users still prefer chronological order and not all of them accept ranking by relevance, Ramarao et al. [25] designed a system called InLook that has an intent panel that shows three results based on relevance as well as a recency panel that shows the rest results based on recency. They further studied the benefit of each part of the system. Carmel et al. [10] proposed to combine h relevance results and k-h time-sorted results in a fixed display window of size k. Then they investigated the performance of allowing or forbidding duplication in the two sets of results and using a fixed or variable h. Mackenzie et al. [20] observed that time-based ranking begins to fail as email age increases, suggesting that hybrid ranking approaches with both relevance-based rankers and the traditional time-based rankers could be a better choice. Nowadays, most email service providers have adopted the strategy of two separate panels for relevance and recency based results and allowing duplicates.

Efforts have also been made to improve the search quality of relevance-based results for the relevance panel. Aberdeen et al.

---

[1]lists.w3.org

[2]http://www.cs.cmu.edu/~enron/

[3]https://ropercenter.cornell.edu/ipoll/study/31086601

[2] exploited social features that capture the degree of interaction between the sender and the recipient for search in Gmail Priority Inbox. Bendersky et al. [5] proposed an attribution parameterization method that uses the click-through rate of frequent n-grams of queries and documents for email ranking. Kuzi et al. [15] studied query expansion methods for email search by using a translation model based on clicked messages, finding neighbor terms based on word embedding models trained with each user's mailbox, and extracting terms from top-ranked results in the initial retrieval with pseudo relevance feedback techniques. Li et al. [16] proposed a multi-view embedding based synonym extraction method to expand queries for email search. Zamani et al. [38] incorporated the situational context such as geographical (country, language) and temporal (weekday, hour) features to facilitate personal search. Shen et al. [27] clustered queries based on the frequent n-grams of results retrieved with a baseline ranker and then used the query cluster information as an auxiliary objective in multi-task learning. There has also been research that studies how to combine sparse and dense features in a unified model effectively [22], combine textual information from queries and documents with other side information to conduct effective and efficient learning to rank [24], and transfer models learned from one domain to another in email search [30].

To the best of our knowledge, there has been no previous work on characterizing different users' preferred ranking signals with users' search history as query context and letting the ranking model adapt to the context, especially based on numerical features rather than terms or n-grams of the queries and email documents as most previous work uses.

Context-aware Ranking in Web Search. Context-aware ranking has been explored extensively in Web search, which is usually based on users' long or short-term search history [6, 13, 17, 18, 21, 28, 31, 35, 36]. Short-term search history includes the queries issued in the current search session and their associated clicked results. Shen et al. [28] proposed a context-aware language modeling approach that can extract expansion terms based on short-term search history and showed the effectiveness of the model on TREC collections. Afterward, some studies also focused on leveraging users' short-term history as context, especially their clicked documents [17, 31, 35, 36]. Long-term search history is not limited to the search behaviors in the current session and can include more users' historical information. It is often used for search personalization [6, 13, 21].

Although we also leverage users' search history to characterize users, we focus on differentiating the importance of various ranking signals for different users rather than refining the semantic intent of a user query with long or short-term context as in Web search. More importantly, we aim to study the effectiveness of search history in the context of email search, which is significantly different from Web search in terms of privacy concerns and the property of search logs.

3 METHOD

In this section, we first describe the problem formulation and available ranking features. Then we propose two neural models, a) a vanilla neural ranking model without using context (NRM) and b)

Table 1: Representative Features in Personal Email Search. Note that no word or n-gram information is available.

Feature Group	Type	Notation	Examples
query-level	discrete	$F_Q^{\mathcal{D}}$	query language, user type
	continuous	F_Q^C	IDF of query terms
document-level	discrete	$F_D^{\mathcal{D}}$	flagged, read, meta-field length
	continuous	F_D^C	recency, email length
q-d matching	continuous	F_{QD}^C	BM25, language model scores

a context-dependent neural ranking model (CNRM) that encodes users' historical behaviors as query context. Afterward, we introduce how we optimize these neural models. At last, we illustrate how to incorporate the query context information with Lambdar-Mart models.

3.1 Background and Problem Formulation

Let q be a query issued by user u_q, D be the set of candidate documents [4] for q which can be obtained from the retrieval model at an early stage. D consists of user u_q's clicked documents D^+ and the other competing documents D^-, i.e., $D = D^+ \cup D^-$. The objective of email ranking is to promote D^+ to the top so that users can find their target documents as soon as possible.

Email Ranking Features. For any document $d \in D$, three groups of features are extracted from a specifically designed privacy-preserved system. Table 1 shows some representative features of each group. (*i*) *Query-level features* $F_Q(q)$ indicate properties of a user's query and the user's mailbox, such as a user type representing whether u_q is a consumer (general public) or a commercial user (employee of a commercial company to which the email service is provided). F_Q can be further divided to discrete features $F_Q^{\mathcal{D}}$ and continuous features F_Q^C, i.e., $F_Q(q) = F_Q^{\mathcal{D}}(q) \cup F_Q^C(q)$. (*ii*) *Document-level features* $F_D(d)$ characterize properties of d such as its recency and size, independent of q. F_D also consists of discrete $F_D^{\mathcal{D}}$ and continuous features F_D^C, i.e., $F_D(d) = F_D^{\mathcal{D}}(d) \cup F_D^C(d)$. (*iii*) *query-document (q-d) matching features* $F_{QD}(q, d)$ measure how d matches the query q in terms of overall and each field in an email, such as "to" or "cc"list, and subject. The *q-d* matching features $F_{QD}(q, d)$ are all considered as continuous features, i.e., $F_{QD}(q, d) = F_{QD}^C(q, d)$. The score of d given q can be computed based on $F_Q(q), F_D(d)$, and $F_{QD}(q, d)$. To protect user privacy, raw text of queries and documents are not available during our investigation.

User Search History. When we further consider user behavior history for the current query q_0 issued by u_{q_0}, more information is available to rank documents in the candidate set D_0. Let $(q_k, q_{k-1}, \cdots, q_1)$ be the recent k queries issued by u_{q_0} before q_0 and $(D_k^+, D_{k-1}^+, \cdots, D_1^+)$ be their associated relevant documents. The features of the user's historical positive q-d pairs are used to identify the user's matching patterns and facilitate the ranking for q_0.

---

[4]We will use document and email interchangeably in this paper.

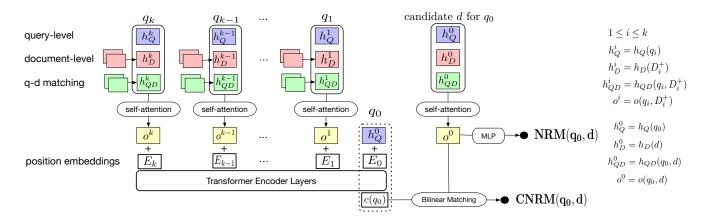Keping Bi[1], Pavel Metrikov[2,4], Chunyuan Li[3], Byungki Byun[2]

Figure 1: The model architecture of the vanilla neural ranking model (NRM) and the context-dependent neural ranking model (CNRM). NRM only uses the features of a candidate d given q_0 to compute its ranking score, while CNRM considers extra context information encoded from the features corresponding to $[q_0, ..., q_k]$ and their associated clicked documents.

3.2 Vanilla Neural Ranking Model

We first describe a baseline method: the vanilla neural ranking model (NRM) only uses the three available groups of ranking features of the current query-document (q-d) pair for ranking without considering user search history. NRM first encodes each group of features to the latent space, then aggregates the hidden vectors of each group of features with a self-attention mechanism. At last, it scores the q-d pair based on the aggregated representation. The model architecture can be found in Figure 1. We will introduce each component of the model next.

Feature Encoding. The three groups of email features have different scales; thus it is important to normalize and map them to the same latent space to maximize the information the model can learn. Discrete features, such as user type, query language, and whether the email has be read, will be projected to hidden vectors by referring to the corresponding embedding lookup tables. Continuous features such as email length and recency will be considered as numerical input vectors directly. Specifically, take query-level features for instance, discrete features $F_Q^{\mathcal{D}}(q)$ will be mapped to

$$g_Q^{\mathcal{D}}(q) = [emb(f_1)^T; \cdots; emb(f_{|F_Q^{\mathcal{D}}(q)|})^T]^T \tag{1}$$

where $[\cdot; \cdot; \cdot]$ denotes the concatenation of a list of vectors, $|\cdot|$ indicates the dimension of a vector, and $emb(f) \in \mathbb{R}^{e \times 1}$ is the embedding of feature f with dimension e. Then the concatenated vector is mapped to $h_Q^{\mathcal{D}} \in \mathbb{R}^{m \times 1}$ with matrix $W_Q^{\mathcal{D}} \in \mathbb{R}^{m \times |g_Q^{\mathcal{D}}(q)|}$, where m is the dimension of the hidden vector $h_Q^{\mathcal{D}}$, according to

$$h_Q^{\mathcal{D}}(q) = \tanh(W_Q^{\mathcal{D}} g_Q^{\mathcal{D}}(q)). \tag{2}$$

Similarly, the continuous query-level features $F_Q^C(q)$ are mapped to $h_Q^C \in \mathbb{R}^{m \times 1}$ with matrix $W_q^C \in \mathbb{R}^{m \times |F_Q^C(q)|}$, i.e.,

$$h_Q^C(q) = \tanh(W_Q^C g_Q^C(q)) \tag{3}$$

Then the two hidden vectors are concatenated as the representation of query-level features of q, denoted as $h_Q(q) \in \mathbb{R}^{2m \times 1}$, namely,

$$h_Q(q) = [h_Q^{\mathcal{D}}(q)^T; h_Q^C(q)^T]^T \tag{4}$$

Likewise, the hidden vector corresponding to the document-level features $F_D(d)$ can be obtained as $h_D(d) = [h_D^{\mathcal{D}}(d)^T; h_D^C(d)^T]^T \in \mathbb{R}^{2m \times 1}$ following the same encoding paradigm but with different lookup tables for discrete features and mapping matrices in the above equations.

Since all the q-d matching features $F_{QD}(q, d)$ are continuous features, they are mapped to

$$h_{QD}(q, d) = h_{QD}^C(q, d) = \tanh(W_{QD}^C F_{QD}(q, d)) \tag{5}$$

where $W_{QD}^C \in \mathbb{R}^{2m \times |F_{QD}(q,d)|}$. Note that the bias terms in all the mapping functions throughout the paper are omitted for simplicity.

Feature Aggregation with Self-attention. To better balance the importance of each group of features in the ranking process, the hidden vectors corresponding to the query, document, and q-d matching features are aggregated according to a self-attention mechanism as follows. [5] They are first mapped to another vector with $W_a \in \mathbb{R}^{2m \times 2m}$:

$$o_Q(q) = \tanh(W_a h_Q(q));$$
$$o_D(d) = \tanh(W_a h_D(d)); \tag{6}$$
$$o_{QD}(q, d) = \tanh(W_a h_{QD}(q, d))$$

$o_Q(q), o_D(d)$, and $o_{QD}(q, d)$ are all in $\mathbb{R}^{2m \times 1}$. Then the attention weight of each component is computed based on softmax function of its dot-product with itself and the other two components. The new vector of each component is the weighted combination of the input and average pooling is applied to the new vectors to obtain the final vector. In concrete, the aggregated vector $o(q, d) \in \mathbb{R}^{2m \times 1}$

---

[5]We also tried to aggregate the vectors of the document and q-d matching features using their attention weights with respect to the vector of the query features. However, this way is not better than using self-attention so we do not include this in the paper.

is computed according to

$$o_c = [o_Q(q); o_D(d); o_{QD}(q,d)]$$

$$W_{attn} = \text{softmax}(o_c^T o_c) \tag{7}$$

$$o(q,d) = \text{AvgPool}(o_c W_{attn})$$

Scoring Function. The score of document d given q is finally computed as:

$$NRM(q,d) = W_{s2} \tanh(W_{s1} o(q,d)) \tag{8}$$

where $W_{s1} \in \mathbb{R}^{m \times 2m}$ and $W_{s2} \in R^{1 \times m}$.

3.3 Context-dependent Neural Ranking Model

A user's historical search behaviors could help characterize the user and provide a query-level context to the current query. We propose a context-dependent neural ranking model (CNRM) to extract a context vector based on the user's search history and scores a candidate document according to both its features given the current query and the query context. We first introduce how CNRM encodes a candidate document's features and then show how CNRM obtains the context vector and scores the candidate document. The model architecture is shown in Figure 1.

Given the current query q_0, for each d in the candidate set D, CNRM obtains the final hidden vector $o(q_0, d)$ from its initial three groups of features following the same paradigm in Section 3.2.

Context Encoding. As shown in Figure 1, for each historical query $q_i (1 \le i \le k)$, we obtain the hidden vector of its query-level features, i.e., $h_Q(q_i)$ using the same encoding process as the current query q_0 according to Equation (1), (2), and (3). Similarly, for each document d_i in the relevant document set D_i^+ of q_i, we encode its document-level features and q-d matching features to the latent space using the same mapping functions and parameters that are used for the candidate document d given q_0, which can be denoted as $h_D(d_i)$ and $h_{QD}(q_i, d_i)$ respectively. Then the aggregated document-level and q-d matching vectors corresponding to D_i^+ are computed as the projected average embeddings of each d_i in D_i^+:

$$h_D(D_i^+) = \tanh(W_d \text{Avg}(\{h_D(d_i) | d_i \in D_i^+\}))$$

$$h_{QD}(q_i, D_i^+) = \tanh(W_{qd} \text{Avg}(\{h_{QD}(q_i, d_i) | d_i \in D_i^+\})), \tag{9}$$

where $W_d \in \mathbb{R}^{2m \times 2m}$ and $W_{qd} \in \mathbb{R}^{2m \times 2m}$. The overall vector corresponding to q_i, i.e., $o(q_i, D_i^+)$, is computed according to Equation (6) and (7) with $h_D(d)$ and $h_{QD}(q, d)$ replaced by $h_D(D_i^+)$ and $h_{QD}(q_i, D_i^+)$ respectively.

To produce a context vector for q_0, we encode the sequence of the overall vectors corresponding to each historical query and the associated positive documents together with the hidden vector of the query-level feature of q_0 with a transformer [33] architecture. Specifically,

$$c(q_0) = \text{Transformer}(o(q_k, D_k^+) + posEmb(k), \cdots, \\ o(q_1, D_1^+) + posEmb(1), h_Q(q_0) + posEmb(0)) \tag{10}$$

As in Figure 1, the output of Transformer [33] is the vector corresponding to q_0 in the final transformer layer, which acts as the final context vector. [6] As such, the interaction between the current

---

[6]We do not introduce how Transformers work due to the space limitations. Please refer to [33] for details.

query-level features and the historical behaviors can be captured and used to better balance the importance of each search behavior in the history.

Scoring Function. To allow each dimension of the encoded features of a candidate d and the context vector interact sufficiently, the final score of d given q_0 is computed by bilinear matching between the aggregated vector of d and the encoded context:

$$CNRM(q_0, d) = o(q_0, d)^T W_b c(q_0) \tag{11}$$

where $W_b \in \mathbb{R}^{2m \times 2m}$, $o(q_0, d)$ and $c(q_0)$ are computed according to Equation (4) and (10) respectively.

3.4 Model Optimization

We use the softmax cross-entropy list-wise loss [3] to train NRM and CNRM. Specifically, the cross-entropy between the list of document labels $y(d)$ and the list of probabilities of each document obtained with the softmax function applied on their ranking scores are used as the loss function. For a query q with candidate set D, the ranking loss $\mathcal{L}_\nabla$ can be computed as:

$$\mathcal{L}_\nabla = -\sum_{d \in D} y(d) \log \frac{\exp(s_r(d))}{\sum_{d' \in D} \exp(s_r(d'))}, \tag{12}$$

where s_r is a scoring function for NRM in Equation (8) or $CNRM$ and in Equation (11). However, the document label $y(d)$ is extracted based on user clicks and thus biased towards top documents since users usually examine the results from top to bottom and decide whether to click. To counteract the effect of position bias, better balance the feature weights, and let context take a properer effect, we train NRM and CNRM through unbiased learning. In contrast to Web search, where there is one panel for organic results, email search usually has two panels with several results ranked by relevance followed by results ranked by time. As stated in [10], allowing duplicates in the two panels leads to better user satisfaction. So each document d can have two positions in terms of the relevance and time window respectively, denoted as $pos_r(d)$, $pos_t(d)$. We adapt the unbiased learning algorithm proposed by Ai et al. [4] to our email search scenario. An examination model is built to estimate the propensity of different positions and adjust each document's weight in the ranking loss with inverse propensity weighting. Meanwhile, the ranking scores can adjust the document weights in the examination loss by inverse relevance weighting as well [4]. We will introduce the examination model and the collaborative model optimization next.

Examination Model. Two lookup tables are created for relevance and time positions respectively. Let $emb(pos_r(d)) \in \mathbb{R}^{m \times 1}$ and $emb(pos_t(d)) \in \mathbb{R}^{m \times 1}$ be the embeddings of relevance and time positions, $W_{p1} \in \mathbb{R}^{m \times 2m}$ and $W_{p2} \in \mathbb{R}^{1 \times m}$ be the mapping matrices. The score of d, i.e., $s_e(d)$, output by the examination model is computed as:

$$h_{p1}(d) = [emb(pos_r(d))^T; emb(pos_t(d)^T]^T$$

$$s_e(d) = W_{p2} \tanh(W_{p1} h_1) \tag{13}$$

Collaborative Optimization. We follow the dual learning algorithm in [4] and use the softmax cross-entropy list-wise loss [3] to train both the examination model and the ranking model. Let d_{r1} be the document that is ranked to the first position in the relevance panel, namely, $pos_r(d_{r1}) = 1$. Inverse propensity weighting based

Keping Bi[1], Pavel Metrikov[2,4], Chunyuan Li[3], Byungki Byun[2]

on the relative propensity compared with the first result is applied to each document and adjust the final loss. Then Equation (12) can be refined as:

$$g_e(d) = \frac{\exp(s_e(d))}{\sum_{d' \in D} \exp(s_e(d'))}$$

$$\mathcal{L}_r = -\sum_{d \in D} y(d) \frac{g_e(d_{r1})}{g_e(d)} \log \frac{\exp(s_r(d))}{\sum_{d' \in D} \exp(s_r(d'))} \quad (14)$$

Regularization terms in Equation (14) has been omitted for simplicity. Similarly, the loss of the examination model can be derived by swapping s_r and s_e in Equation (14). The training of the examination and ranking models depends on each other and both of them are refined step by step until the models converge.

3.5 Incorporate Context to LambdaMart

It is hard to directly incorporate user behavior history into some learning-to-rank (LTR) models, such as the state-of-the-art method – LambdaMart [8], since it is grounded on various ready-to-use features and does not support learning features automatically during training. To study whether user search history can characterize the context of a query q and benefit search quality for LambdaMart, we investigate a method to incorporate the context vector encoded from user search history, i.e., $c(q)$ in Equation (10), to the LambdaMart model.

We first cluster the context vectors to n clusters with k-means [7] and obtain a one-hot vector of length n, denoted as $cluster(q) \in \mathbb{R}^{n \times 1}$. The dimension of the corresponding cluster id of the context has value 1 and the rest dimensions are all 0. To let the context take more effect, instead of adding $cluster(q)$ as features to a LambdaMart model directly, we bundle the document and q-d matching features with the $cluster(q)$. Specifically, we extract l most important features in the feature set $[F_D(d); F_{QD}(q_0, d)]$, denoted as $F_l(q, d) \in \mathbb{R}^{l \times 1}$, do $F_l(q, d) \cdot cluster(q)^T$, flatten the yielded matrix to a vector F_{fc} of length $l * n$, and add F_{fc} to LambdaMart as additional features. In our experiments, l is set to 3 and $F_l(q, d)$ includes recency, email length, and the overall BM25 score computed with multiple fields (BM25f) [26].

We also tried other methods of incorporating the context into LambdaMart, such as adding the context vector as $2m$ features directly, adding the cluster id as a single feature, and adding n features either with a one-hot vector according to the cluster id or the probability of the context belonging to each cluster. These approaches are worse than the method we propose so we do not include the corresponding experimental results in the paper.

4 EXPERIMENTAL SETUP

4.1 Data & Privacy Statement

We use search logs obtained from one of the world's largest email search engines to collect the data. To protect user privacy, the raw text of their queries and documents cannot be accessed. Only features extracted from specifically designed privacy-preserved pipelines are available, as mentioned in Section 3. Each candidate document has over 300 associated features. We randomly sampled some users from two-week search logs and only kept the users who

have issued more than 10 queries during this period. We observe that over 10% of users satisfy this condition. To reduce the effect of the uncommon users with too many queries, we filtered out 90% of the users who issued more than 20 queries. In this way, we obtained hundreds of thousands of queries associated with the users in total. There are about 30 candidate documents on average and at most 50 documents for each query, out of which about 3 documents have positive labels on average. The labels are extracted based on users' interaction with the documents, such as click, reply, forward, long-dwell, etc., and there are five levels in total - bad, fair, good, excellent, and perfect. At each level, the average number of documents given a query is 26.91, 1.87, 0.03, 0.55, and 0.64, respectively.

Note that the dataset we produce has no user content, and it is also anonymized and delinked from the end-user. Under the End-user License Agreement (EULA), the dataset can be used to improve the service quality.

Data Partition. To obtain data partitions consistent with real scenarios, we ordered the queries by their search time and divided the data to the training/validation/test set according to 0.8:0.1:0.1 in chronological order. Experiments on both neural models and LambdaMart models are based on this data.

To see whether CNRM can generalize well to unseen users and achieve better search quality based on numerical ranking features in their search history, we also partition the data according to users. Specifically, we randomly split all the users to training/validation/test according to 0.8:0.1:0.1 and put their associated queries to the corresponding partitions. The users in each partition do not overlap.

4.2 Evaluation

We conduct two types of evaluation to see whether users' historical behaviors benefit search quality. Specifically, we compare NRM to CNRM and the vanilla LambdaMart model to the LambdaMart model with context added as additional features (as stated in Section 3.5). These comparisons are based on the data partitioned in chronological order, which is consistent with real scenarios. Also, we compare CNRM with NRM on data partitioned according to users to show the ability of CNRM generalizing to unseen users. When we conduct comparisons between neural models with and without incorporating the context, we also include several other baselines:

- **BM25f**: BM25f measures the relevance of an email. It computes the overall BM25 score of an email based on word matching in its multiple fields [12, 26].
- **Recency**: Recency ranks emails according to the time they were created, which is the ranking criterion for the recency panel of most email search engines.
- **CLSTM**: The Context-aware Long Short Term Memory model (CLSTM) encodes the context of user behavior features with LSTM [14] instead of Transformers [33]. In other words, CLSTM replaces the Transformers in CNRM with LSTM.

Since top results matter more when users examine the emails and our data has multi-grade labels, we use Normalized Discounted Cumulative Gain (NDCG) at cutoff 3,5,10 as evaluation metrics. Two-tailed student's t-tests with p<0.05 are conducted to check statistical significance.

---

[7]Other clustering methods can be used as well. We leave it as future work.

4.3 Training Settings

We implemented the neural models with PyTorch [8] and trained each neural model on a single NVIDIA Tesla K80 GPU. All the neural models were trained with 128 as batch size and for 10 epochs, within which they can converge well. We set embedding size of discrete features $e = 10$ (in Equation (1)) and the hidden dimension m in Section 3 to 64. The performance does not improve with larger embedding sizes so we only report results with $m = 64$. For the Transformers in CNRM, we sweep the number of attention heads from {1,2,4,8}, the transformer layers from 1 to 3, and the dimension size of the feed-forward sub-layer from {128, 256, 512}. For the CLSTM model, we tried layer numbers from 1 to 3. We set the history length k to 10. We use Adam with $\beta_1 = 0.9, \beta_2 = 0.999$ to optimize the neural models. The learning rate is initially set to 0.002 and then warmed up over the first {2,000, 3,000, 4,000} steps, following the paradigm in [33]. The coefficient for regularization Equation (14) is selected from {1e-5, 5e-5, 1e-6}.

We train the LambdaMart models using LightGBM [9]. We set the number of trees to be 500, the number of leaves in each tree as 150, the shrinkage rate as 0.3, the objective as LambdaRank, and the rounds of early-stop based on the validation performance as 30. We use default parameters for the other settings. The number of clusters n is set to 10.

5 RESULTS AND DISCUSSION

In this section, we first show the experimental results of neural models, conduct ablation studies, and analyze what information the context vector captures. Then we show the results of LambdaMart models with the context information added and give some analysis accordingly.

5.1 Neural Model Evaluation

Overall Performance. Table 2 shows that CNRM and CLSTM outperform NRM on both versions of data, i.e., partitioned according to time or users. This indicates that encoding the ranking features corresponding to users' historical queries and positive documents can benefit the neural ranking model. This information improves search quality not only for the future queries of the same user but also for an unseen user's queries given his/her previous queries. CNRM has better performance than CLSTM, showing the superiority of Transformers over the recurrent neural networks, similar to the observation in [33].

BM25f and Recency both have significantly degraded performance compared to NRM. It is not surprising since they are used together with other features in the neural models. Also, Recency outperforms BM25f by a large margin. In contrast to Web search where relevance usually matters the most, recency plays a more crucial role than relevance in email search. This observation is consistent with the fact that recency has been the criterion for ranking emails in traditional email search engines and the recency panel in most current email search engines.

NRM has worse NDCG@3,5,10 but better NDCG@1 than the vanilla LambdaMart model on both versions of data. We have not included the performance of vanilla LambdaMart in Table 2 since

Table 2: Comparisons between CNRM and the baselines on the data partitioned according to time and users. The reported numbers are the improvements over NRM. '*' and '†' indicate statistically significant improvements over NRM and CLSTM respectively.

Partition	Method	NDCG@3	NDCG@5	NDCG@10
Time	BM25f	-51.55%	-46.95%	-41.37%
	Recency	-38.32%	-32.27%	-26.69%
	NRM	+0.00%	+0.00%	+0.00%
	CLSTM	+1.29%*	+1.35%*	+1.51%*
	CNRM	**+2.52%*†**	**+2.89%*†**	**+2.88%*†**
Users	BM25f	-51.17%	-46.61%	-41.37%
	Recency	-37.23%	-31.58%	-25.94%
	NRM	+0.00%	+0.00%	+0.00%
	CLSTM	+1.79%*	+2.15%*	+2.28%*
	CNRM	**+2.22%*†**	**+2.60%*†**	**+2.75%*†**

Table 3: Performance improvements of the variants of CNRM over NRM. F_Q, F_D, and F_{QD} indicates the CNRM with only query, document, and q-d matching features encoded. * indicates statistically significant differences with NRM.

Model	NDCG@3	NDCG@5	NDCG@10
Full CNRM	**+2.52%***	**+2.89%***	**+2.88%***
with F_Q	+1.12%*	+1.19%*	+1.11%*
with F_D	+1.71%*	+2.03%*	+2.17%*
with F_{QD}	+1.95%*	+2.25%*	+2.25%*
with $F_D \& F_{QD}$	+2.14%*	+2.49%*	+2.56%*
w/o posEmb	+0.72%*	+0.99%*	+1.15%*

Table 2 is for comparison between heuristic methods and neural models. NRM has 6.65% better NDCG@1, while 7.99%, 8.90%, and 7.78% worse NDCG@3,5, and 10, than LambdaMart on the data partitioned according to time. On the data partitioned according to users, NRM has 7.40% better NDCG@1 but 5.02%, 5.42%, and 4.53% worse NDCG@3,5,10 than vanilla LambdaMart. With the same feature set, the advantages of LambdaMart, such as the gradient boosting mechanism, the NDCG-aware loss function, and the ability to handle continuous features, make it hard to beat by a neural model.

Ablation Study. We conduct ablation studies on the data partitioned according to time by keeping only some groups of features and remove the others when encoding the context to study their contributions. In other words, we only use the query features F_Q, the document features F_D, the q-d matching features F_{QD}, or both F_D and F_{QD} corresponding to the historical queries and associated positive documents during context encoding. The current query features are also excluded during encoding (See Figure 1) so that the differences between CNRM and NRM only come from the encoding of certain groups of features. As shown in Table 3, each group of feature alone has a contribution, among which q-d matching features contribute the most and query features contribute the least. It is consistent with our intuition since query features indicate the property of the query or the user while document and q-d matching

Keping Bi[1], Pavel Metrikov[2,4], Chunyuan Li[3], Byungki Byun[2]

(a) Context vectors encoded with F_{QD} w.r.t. user type

(b) Context vectors encoded with F_D and F_{QD} w.r.t. user type

(c) Context vectors encoded with F_D w.r.t. user type

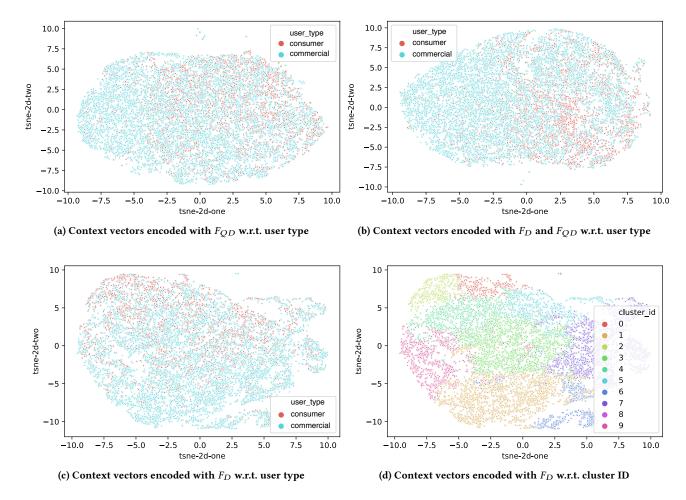(d) Context vectors encoded with F_D w.r.t. cluster ID

Figure 2: 2-D visualization of context vectors in CNRM by t-SNE [32] with respect to user_type and cluster ID.

features carry the most critical information to determine whether a document is a target or not, such as recency and relevance. Incorporating both F_D and F_{QD} leads to better performance while still worse than full CNRM with all the components.

We also remove the position embeddings before feeding the sequence to transformer layers to see whether the ordering of the sequence matters. Results show that the performance will drop significantly without position embeddings, indicating that the ordering of users' historical behaviors is important during encoding.

In addition, without unbiased learning, for both NRM and CNRM, the performance will degrade by about 15% of the models using unbiased learning. The improvement percentage of CNRM over NRM is similar to the numbers shown in Table 3.

Context Vector Analysis. To see whether the context vector $c(q_0)$ (in Figure 1) captures some information that can differentiate users, we try to visualize $c(q_0)$ of each query in terms of a user type. As we mentioned in Section 1, consumer and commercial users have significantly different behavioral patterns. We want to check whether the differences can be revealed when CNRM only encodes document features (F_D), q-d matching features (F_{QD}), or both of them. Meanwhile, the query-features of the current query q_0 in Figure 1 is replaced by a trainable vector of dimension $2m$. This ensures that all query features that include the user type (consumer or commercial) are excluded during encoding and $c(q_0)$ has not seen the user type in advance.

Based on the data partitioned according to time, we use t-SNE [32], which can reserve both the local and global structures of the high-dimensional data well when mapping to low-dimensional space as discussed in [32], to map $c(q_0)$ to the 2-d plane. We randomly sample 10,000 queries and obtain their corresponding context vectors with the same CNRMs that are evaluated in Table 3. Figure 2a, 2b, and 2c shows the 2-d visualization of context vectors encoded with F_{QD}, both F_{QD} and F_D, and F_D in terms of user type respectively.

We observe that by encoding F_{QD}/F_D alone, $c(q_0)$ can differentiate commercial and consumer users to some extent by putting most of consumer users to the right/upper-left part of Figure 2a/2c. With both F_D and F_{QD} encoded, $c(q_0)$ further pushes consumer users to the right part of Figure 2b. These observations indicate that the context vectors of CNRM can differentiate consumer and commercial

users by encoding the document and q-d matching features of their positive historical documents. The two groups of features take a decent effect individually to capture their different behavioral patterns and are more discriminative when used together. It is worth mentioning that the context vectors have captured information beyond whether the user is a commercial or consumer user. This can be verified by the fact that the user type of the current query (q_0) is already used in the candidate document for ranking and the learned context still leads to significantly better performance, as shown in Section 5.1.

5.2 LambdaMart Model Evaluation

In common practice, LambdaMart has been widely used to aggregate multiple features to serve online query requests due to its state-of-the-art performance and high efficiency. So it is very important to find out whether the context information learned in CNRM can benefit the LambdarMart model. The baseline LambdaMart model was trained with the same feature set mentioned in Section 3. For any version of CNRM to be evaluated, the context vector for each query was extracted and put to one of the 10 clusters; the top 3 features (Recency, EmailLength, and BM25f) are bundled with each cluster and added to LambdaMart as additional features, as stated in Section 3.5. Then a new LambdarMart model is trained and evaluated against the baseline version.

Overall Performance. Based on the data partitioned according to time (See Section 4.1), the context information from four versions of CNRM is introduced to the LambdarMart model, which are CNRM that only encodes document features (F_D), q-d matching features (F_{QD}), both F_D and F_{QD}, and full information in the context, same as the settings in our ablation study in Section 5.1. We did not include the context encoded with query features F_Q alone since user-level information such as user type is the same for all user queries and contains less extra information than F_D and F_{QD}.

Since the clustering process forces the representations to degrade from dense to discrete, much information could be lost. In addition, query-level features usually take less effect than document-level or q-d matching features since they are the same for all the candidate documents under a query. Thus, it is challenging to make the query context work effectively in the LambdaMart model. Nevertheless, as shown in Table 4, the context learned from CNRM that encodes F_D alone leads to significantly better performance than the baseline LambdaMart model, which is also the best performance among all.

From Table 4, we observe that the improvements each method obtains in the LambdaMart model do not follow the same trends of their improvements over the baseline NRM shown in Table 3. This is not surprising since the clustering process could lose some information and the condensed query cluster information may have uncertain overlap with existing user properties such as user type and mailbox locale, which already take effect in LambdaMart.

Feature Weight Analysis. In the LambdaMart models, the top 3 features with the most feature weight are Recency, BM25f, and EmailLength, which occupy about 24%, 5%, and 4% of the total feature weights. These features could play a different role in terms of different query clusters. To compare the weights of these 3 features when bundled with each cluster, we plot the feature weights of the LambdaMart model that has significant improvements over the

Table 4: Performance improvements of the LambdaMart model with the context cluster information over the baseline version without context on the data partitioned according to time. * indicates statistically significant differences.

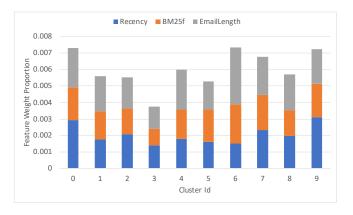| Model	NDCG@3	NDCG@5	NDCG@10
+ Context with F_D	+0.60%*	+0.49%*	+0.56%*
+ Context with F_{QD}	+0.34%	+0.35%	+0.28%
+ Context with $F_D\&F_{QD}$	+0.33%	+0.41%	+0.30%
+ Context from Full CNRM	+0.37%	+0.20%	+0.18%

Figure 3: Feature weights of the added features (Recency, EmailLength, and BM25f) in the LambdaMart model that uses context encoded with F_D in terms of cluster ID.

baseline, which is incorporating the context encoded with F_D alone, in Figure 3. Cluster 6 has the most overall feature weights and has a significantly different feature weight distribution than the other clusters. Queries in Cluster 6 emphasize email length the most and recency the least, which indicates that this cluster requires different matching patterns from the global distribution. As shown in Figure 2d, Cluster 6 is located in the lower-right corner of the figure, which has a little gap with the other points. This may also indicate that Cluster 6 is quite different from the other clusters.

User Distribution w.r.t. Their Query Clusters. In this part, we study how users are distributed in terms of the clusters their queries belong to. We aim to answer the following question: do the query contexts associated with the same user concentrate or scatter in the latent space? We use two criteria to analyze the corresponding user distribution: (1) the number of unique clusters a user's queries belong to, (2) the entropy of the cluster distribution of queries associated with a user. The entropy criterion differentiates query counts in each cluster while the cluster count criterion does not. Then we divide the possible values of cluster count and entropy into 10 bins and put users into each bin to see the distribution. Figure 4 shows the user distribution in percentage regarding the two criteria using the CNRM that only encodes F_D in the context.

We have not observed any pronounced patterns from the user distribution. The two resulting distributions are similar. Most users have 4 or 5 query clusters (Bin 3 or 4) and fall into Bin 5 or 6 in terms of entropy. Most users do not have concentrated query clusters, indicating that user queries could have dynamic context along with

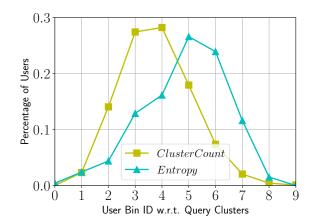Keping Bi[1], Pavel Metrikov[2,4], Chunyuan Li[3], Byungki Byun[2]

Figure 4: The distribution of users when we put each user to 10 bins according to the count of unique clusters the user's queries belongs to and the entropy of the user's query cluster distribution.

time. This makes sense since the context learned in CNRM is related to not only the users' static properties but also their issued queries that could be diversified. We leave the study of extracting users' static properties and the effect of the user cluster information as our future work.

6 CONCLUSION AND FUTURE WORK

This paper presents the first comprehensive work to improve the email search quality based on ranking features in the user search history. We investigate this problem on both neural models and the state-of-the-art learning-to-rank model - LambdaMart [8]. We leverage the ranking features of users' historical queries and the corresponding positive emails to characterize the users and propose a context-dependent neural ranking model (CNRM) that encodes the sequence of the ranking features with a transformer architecture. For the LambdaMart model, we cluster the query context vectors obtained from CNRM and incorporate the cluster information into LambdaMart. Based on the dataset constructed from the search log of one of the world's largest email search engines, we show that for both neural models and the LambdaMart model, incorporating this query context could improve the search quality significantly.

As a next step, we would like to investigate using another optimization goal in addition to the current ranking optimization to enhance the neural ranker, which is differentiating the query from the same user and the other users given the query context encoded from a user's search history. We are also interested in studying how to incorporate the query context into LambdaMart more effectively, such as investigating other clustering methods including neural end-to-end clustering approaches and incorporating the context information differently. As mentioned earlier, extracting users' static properties and use them as query context is another interesting direction.

REFERENCES

[1] Samir AbdelRahman, Basma Hassan, and Reem Bahgat. 2010. A new email retrieval ranking approach. *arXiv preprint arXiv:1011.0502* (2010).
[2] Douglas Aberdeen, Ondrey Pacovsky, and Andrew Slater. 2010. The learning behind gmail priority inbox. (2010).
[3] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. *arXiv preprint arXiv:1804.05936* (2018), 135–144.
[4] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. *arXiv preprint arXiv:1804.05938* (2018).
[5] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from user interactions in personal search via attribute parameterization. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 791–799.
[6] Paul N Bennett, Ryen W White, Wei Chu, Susan T Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short-and long-term behavior on search personalization. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 185–194.
[7] Abhijit Bhole and Raghavendra Udupa. 2015. On correcting misspelled queries in email search. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 4266–4267.
[8] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.
[9] David Carmel, Guy Halawi, Liane Lewin-Eytan, Yoelle Maarek, and Ariel Raviv. 2015. Rank by time or by relevance? Revisiting email search. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 283–292.
[10] David Carmel, Liane Lewin-Eytan, Alex Libov, Yoelle Maarek, and Ariel Raviv. 2017. Promoting relevant results in time-ranked mail search. In *Proceedings of the 26th International Conference on World Wide Web*. 1551–1559.
[11] Nick Craswell, Arjen P De Vries, and Ian Soboroff. 2005. Overview of the TREC 2005 Enterprise Track.. In *Trec*, Vol. 5. 1–7.
[12] Nick Craswell, Hugo Zaragoza, and Stephen Robertson. 2005. Microsoft cambridge at trec-14: Enterprise track. (2005).
[13] W Bruce Croft and Xing Wei. 2005. *Context-based topic models for query modification*. Technical Report. CIIR Technical Report, University of Massachusetts.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[15] Saar Kuzi, David Carmel, Alex Libov, and Ariel Raviv. 2017. Query expansion for email search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 849–852.
[16] Cheng Li, Mingyang Zhang, Michael Bendersky, Hongbo Deng, Donald Metzler, and Marc Najork. 2019. Multi-view Embedding-based Synonyms for Email Search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 575–584.
[17] Zhen Liao, Daxin Jiang, Jian Pei, Yalou Huang, Enhong Chen, Huanhuan Cao, and Hang Li. 2013. A vlHMM approach to context-aware search. *ACM Transactions on the Web (TWEB)* 7, 4 (2013), 1–38.
[18] Sam Lobel, Chunyuan Li, Jianfeng Gao, and Lawrence Carin. 2019. RaCT: Toward Amortized Ranking-Critical Training For Collaborative Filtering. In *International Conference on Learning Representations*.
[19] Craig Macdonald and Iadh Ounis. 2006. Combining fields in known-item email search. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 675–676.
[20] Joel Mackenzie, Kshitiz Gupta, Fang Qiao, Ahmed Hassan Awadallah, and Milad Shokouhi. 2019. Exploring user behavior in email re-finding tasks. In *The World Wide Web Conference*. 1245–1255.
[21] Nicolaas Matthijs and Filip Radlinski. 2011. Personalizing web search using long term browsing history. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 25–34.
[22] Yu Meng, Maryam Karimzadehgan, Honglei Zhuang, and Donald Metzler. 2020. Separate and Attend in Personal Email Search. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 429–437.
[23] Paul Ogilvie and Jamie Callan. 2005. Experiments with Language Models for Known-Item Finding of E-mail Messages.. In *TREC*.
[24] Zhen Qin, Zhongliang Li, Michael Bendersky, and Donald Metzler. 2020. Matching Cross Network for Learning to Rank in Personal Search. In *Proceedings of The Web Conference 2020*. 2835–2841.

[25] Pranav Ramarao, Suresh Iyengar, Pushkar Chitnis, Raghavendra Udupa, and Balasubramanyan Ashok. 2016. Inlook: Revisiting email search experience. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1117–1120.

[26] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. 42–49.

[27] Jiaming Shen, Maryam Karimzadehgan, Michael Bendersky, Zhen Qin, and Donald Metzler. 2018. Multi-task learning for email search ranking with auxiliary query clustering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2127–2135.

[28] Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2005. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 824–831.

[29] Ian Soboroff, Arjen P de Vries, and Nick Craswell. 2006. Overview of the TREC 2006 Enterprise Track.. In *Trec*, Vol. 6. 1–20.

[30] Brandon Tran, Maryam Karimzadehgan, Rama Kumar Pasumarthi, Michael Bendersky, and Donald Metzler. 2019. Domain Adaptation for Enterprise Email Search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 25–34.

[31] Yury Ustinovskiy and Pavel Serdyukov. 2013. Personalization of web-search using short-term browsing context. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 1979–1988.

[32] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[34] Wouter Weerkamp, Krisztian Balog, and Maarten De Rijke. 2009. Using contextual information to improve search in email archives. In *European Conference on Information Retrieval*. Springer, 400–411.

[35] Ryen W White, Paul N Bennett, and Susan T Dumais. 2010. Predicting short-term interests using activity-based search context. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. 1009–1018.

[36] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. 2010. Context-aware ranking in web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 451–458.

[37] Sirvan Yahyaei and Christof Monz. 2008. Applying maximum entropy to known-item email retrieval. In *European Conference on Information Retrieval*. Springer, 406–413.

[38] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational context for ranking in personal search. In *Proceedings of the 26th International Conference on World Wide Web*. 1531–1540.