Reply With: Proactive Recommendation of Email Attachments

Christophe Van Gysel*
cvangysel@uva.nl
University of Amsterdam
Amsterdam, The Netherlands

Bhaskar Mitra
Matteo Venanzi
Roy Rosemarin
bmitra@microsoft.com
mavena@microsoft.com
rorosema@microsoft.com
Microsoft
United Kingdom

Grzegorz Kukla
Piotr Grudzien
Nicola Cancedda
grkukla@microsoft.com
a-pigrud@microsoft.com
nicanced@microsoft.com
Microsoft
United Kingdom

ABSTRACT

Email responses often contain items—such as a file or a hyperlink to an external document—that are attached to or included inline in the body of the message. Analysis of an enterprise email corpus reveals that 35% of the time when users include these items as part of their response, the attachable item is already present in their inbox or sent folder. A modern email client can proactively retrieve relevant attachable items from the user's past emails based on the context of the current conversation, and recommend them for inclusion, to reduce the time and effort involved in composing the response. In this paper, we propose a weakly supervised learning framework for recommending attachable items to the user. As email search systems are commonly available, we constrain the recommendation task to formulating effective search queries from the context of the conversations. The query is submitted to an existing IR system to retrieve relevant items for attachment. We also present a novel strategy for generating labels from an email corpus—without the need for manual annotations—that can be used to train and evaluate the query formulation model. In addition, we describe a deep convolutional neural network that demonstrates satisfactory performance on this query formulation task when evaluated on the publicly available Avocado dataset and a proprietary dataset of internal emails obtained through an employee participation program.

1 INTRODUCTION

In spite of the growing popularity of social networks and other modern online communication tools, email is still pervasive in the enterprise space [44]. Users typically respond to incoming emails with textual responses. However, an analysis of the publicly available Avocado dataset [42] reveals that 14% of those messages also contain items, such as a file or a hyperlink to an external document. Popular email clients already detect when users forget to attach files by analyzing the text of the response message [16, 17]. On Avocado, we find that in 35% of the cases where the response contains attachments, the item being attached is also present in the sender's mailbox at the time of composing the response. This implies that modern email clients could help users compose their responses faster by proactively retrieving and recommending relevant items that the user may want to include with their message.

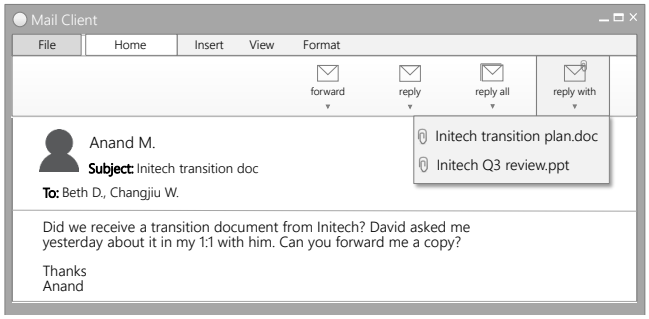


Figure 1: Anand asks Beth and Changjiu to forward him a copy of the Initech[1] transition document. Beth's email client recommends two files, part of earlier emails in Beth's mailbox, for her to attach to her reply.

In *proactive* information retrieval (IR) systems [8, 33, 54, 56], the user does not initiate the search. Instead, retrieval is triggered automatically based on a user's current context. The context may include the time of day [56], the user's geographic location [8], recent online activities [54] or some other criteria. In our scenario, retrieval is based on the context of the current conversation, and in particular, the message the user is responding to. In a typical IR scenario, items are ranked based on query-dependent feature representations. In the absence of an explicit search query from the user, proactive IR models may formulate a keyword-based search query using the available context information and retrieve results for the query using a standard IR model [33]. Search functionalities are available from most commercial email providers and email search has been studied in the literature [1]. Therefore, we cast the email attachment recommendation problem as a query formulation task and use an existing IR system to retrieve emails. Attachable items are extracted from the retrieved emails and a ranking is presented to the user.

Fig. 1 shows an example of an email containing an explicit request for a file. In general, there may or may not be an explicit request, but it may be appropriate to attach a relevant file with the response. Our task is to recommend the correct "transition document" as an attachment when Beth or Changjiu is responding to this email. In order to recommend an attachment, the model should formulate a query, such as "*Initech transition*", based on the context of the

*Work performed while the first author was at Microsoft, London.
[1]Initech is a fictional company from a popular 1990s comedy film. Any resemblance to real organizations is purely coincidental.

CIKM'17 , November 6–10, 2017, Singapore, Singapore

ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132979

request message, that retrieves the correct document from Beth's or Changjiu's mailbox. To formulate an effective query, the model must identify the discriminative terms in the message from Anand that are relevant to the actual file request.

Machine learning models that aim to solve the query formulation task need reliable feedback on what constitutes a good query. One option for generating labeled data for training and evaluation involves collecting manual assessments of proposed queries or individual query terms. However, it is difficult for human annotators to determine the ability of a proposed query to retrieve relevant items given only the request message. In fact, the efficacy of the query depends on the target message that should be retrieved, as well as how the IR system being employed functions. The relevance of the target message, in turn, is determined by whether they include the correct item that should be attached to the response message. Therefore, instead we propose an evaluation framework that requires an email corpus but no manual assessments. Request/response message pairs are extracted from the corpus and the model, that takes the request message as input, is evaluated based on its ability to retrieve the items attached to the response message. An IR system is employed for the message retrieval step, but is treated as a *black box* in the context of evaluation. Our framework provides a concise specification for the email attachment recommendation task (§3).

Our proposed approach for training a deep convolutional neural network (CNN) for the query formulation step is covered in §4. The model predicts a distribution over all the terms in the request message and terms with high predicted probability are selected to form a query. Model training involves generating a dataset of request/attachment pairs similar to the case of evaluation. Candidate queries are algorithmically synthesized for each request/attachment pair such that a message from the user's mailbox with the correct item attached is ranked highly. We refer to synthetic queries as the *silver-standard queries* (or silver queries for brevity) to emphasize that they achieve reasonable performance on the task, but are potentially sub-optimal. The neural model is trained to minimize the prediction loss w.r.t. the silver queries given the request message as input.

The key contributions of this paper are as follows. (1) We introduce a novel proactive retrieval task for recommending email attachments when composing a response to an email message. The task involves formulating a query such that relevant attachable items are retrieved. Towards this goal, an evaluation framework is identified that requires an email corpus, but no manual annotations or assessments. (2) We show how a machine learning model can be trained on an unlabeled email corpus for this query formulation task. The training involves generating a dataset of request/response message pairs and corresponding sets of silver queries. (3) Finally, we present a neural model for estimating a probability distribution over the terms in the request message for formulating queries.

2 RELATED WORK

2.1 Proactive IR

Zero-query search—or proactive IR—scenarios have received increasing attention recently [2]. However, similar approaches have also been studied in the past under other names, such as *just-in-time* [46–49], query-free [23] or anticipatory [10, 33] IR. According to Hart and Graham [23], the goal of the proactive retrieval system is to surface information that helps the user in a broader task. While some of these works focus on displaying contextually relevant information next to Web pages [10, 15, 36, 46, 48] or multimedia [43], others use audio cues [41, 52] or signals from other sensors [47, 50] to trigger the retrieval. In more recent years, proactive IR systems have re-emerged in the form of intelligent assistant applications on mobile devices, such as Siri, Google Now and Cortana. The retrieval in these systems may involve modeling repetitive usage patterns to proactively show concise information cards [54, 56] or surface them in response to change in user context such as location [8]. Hart and Graham [23], Budzik and Hammond [10] and Liebling et al. [33] propose to proactively formulate a query based on user's predicted information need. In contrast to previous work on proactive contextual recommendation, we formulate a query to retrieve attachable items to assist users with composing emails instead of supplying information to support content or triggering information cards in mobile assistants.

2.2 Predictive models for email

Email overload is the inability to effectively manage communication due to the large quantity of incoming messages [61]. Grevet et al. [21] find that work email tends to be overloaded due to outstanding tasks or reference emails saved for future use. Ai et al. [1] find that 85% of email searches are targeted at retrieving known items (e.g., reference emails, attachments) in mailboxes. Horvitz [25] argues that a combination of two approaches—(1) providing the user with powerful tools, and (2) predicting the user's next activity and taking automated actions on her behalf —is effective in many scenarios. Modern email clients may better alleviate email overload and improve user experience by incorporating predictive models that try to anticipate the user's need and act on their behalf.

Missing attachments in email generates a wave of responses notifying the sender of her error. Dredze et al. [16] present a method that notifies the user when a file should be attached before the email is sent. Castro et al. [13] proposed a learning framework to predict the action that will be performed on an email by the user, with the aim of prioritizing actionable emails. Carvalho and Cohen [12] classify emails according to their speech acts. Graus et al. [20], Qadir et al. [45] recommend recipients to send an email message to. Kannan et al. [28] propose an end-to-end method for automatically generating email responses that can be sent by the user with a single click.

2.3 Query formulation and reformulation

The task of contextual recommendation of attachable items by means of query formulation has not received much attention. However, there is work on query extraction from verbose queries and query construction for related patent search. Similar to this paper, the methods below consider the search engine as a black box.

Prior art search. Establishing novelty is an important part of the patenting process. Patent practitioners (e.g., lawyers, patent office examiners) employ a search strategy where they construct a query based on a new patent application in order to find prior art. However, patents are different from typical documents due to their length and lack of mid-frequency terms [34].

Automated query generation methods have been designed to help practitioners search for prior art. Xue and Croft [64] use TF-IDF to generate a ranking of candidate query terms, considering different patent fields, to rank similar patents. In later work [63],

they incorporated a feature combination approach to further improve prior art retrieval performance. Alternative term ranking features, such as relative entropy [37], term frequency and log TF-IDF [14], have also been explored. Kim et al. [29] suggest boolean queries by extracting terms from a pseudo-relevant document set. Golestan Far et al. [19] find that an interactive relevance feedback approach outperforms state-of-the-art automated methods in prior art search.

Query extraction methods used for prior art search can also be applied to the task of attachable item recommendation considered in this paper. Consequently, we consider the methods mentioned above as our baselines (§5.4). However, there are a few notable differences between the patent and email domains: (1) Email messages are much shorter in length than patents. (2) Patents are more structured (e.g., US patents contain more than 50 fields) than email messages. (3) Patents are linked together by a static citation graph that grows slowly, whereas email messages are linked by means of a dynamic conversation that is fast-paced and transient in nature. (4) In the case of email, there is a social graph between email users that can act as an additional source of information.

Improving verbose queries. Bendersky and Croft [6] point out that search engines do not perform well with verbose queries [4]. Kumaran and Carvalho [31] propose a sub-query extraction method that obtains ground truth by considering every sub-query of a verbose query and cast it as a learning to rank problem. Xue et al. [65] use a Conditional Random Field (CRF) to predict whether a term should be included. However, inference using their method becomes intractable in the case of long queries. Lee et al. [32] learn to rank query terms instead of sub-queries with a focus on term dependency. Huston and Croft [26] find that removing the stop structure in collaborative question answering queries increases retrieval performance. Maxwell and Croft [38] propose a method that selects query terms based on a pseudo-relevance feedback document set. Meij et al. [39] identify semantic concepts within queries to suggest query alternatives. Related to the task of improving verbose queries is the identification of important terms [67]. He and Ounis [24] note that the use of relevance scores for query performance prediction is expensive to compute and focus on a set of pre-retrieval features that are strong predictors of the query's ability to retrieve relevant documents. See [22] for an overview.

The task we consider in this paper differs from search sub-query selection as follows. (1) Search queries are formulated by users as a way to interface with a search engine. Requests in emails may be more complex as they are formulated to retrieve information from a human recipient, rather than an automated search engine. In other words, email requests are more likely to contain natural language and figurative speech than search engine queries. This is because the sender of the request does not expect their message to be parsed by an automated system. (2) Search sub-query extraction aims to improve retrieval effectiveness while the query intent remains fixed. This is not necessarily the case in our task, as a request message like has the intent to retrieve information from the recipient (rather than a retrieval system operating on top of the recipient's mailbox). (3) Work on search sub-query selection [31, 65] takes advantage of the fact that 99.9% of search queries consist of 12 terms or less [7] by relying on computations that are intractable otherwise. As emails are longer (Table 2), many of the methods designed for search sub-query selection are not applicable in our setting.

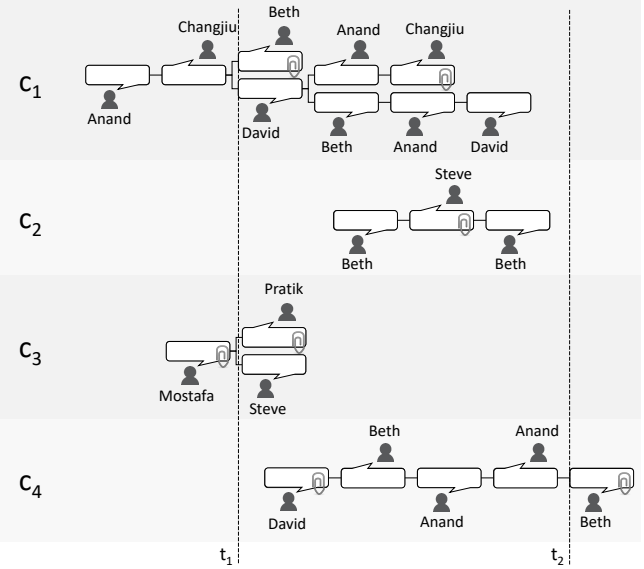


Figure 2: Beth's mailbox has four on-going conversations. When Beth replies with an attachment in conversation c_1, at time t_1, only the attachment she received from Mostafa (conversation c_3) is present in her mailbox.

3 PROACTIVE ATTACHABLE ITEM RECOMMENDATION

Given message $m_{a\to b}$ from user u_a to user u_b, we want to recommend an item e that the receiver u_b may want to attach (or include) in the response $m_{b\to a}$. Email corpora, such as Avocado [42], contain many conversation threads where each conversation c contains messages $m_i \in c$ exchanged between several participants. From these conversations, we can identify pairs of request-response messages $\langle m_{a\to b}, m_{b\to a}\rangle$ where $m_{b\to a}$ contains an attachment e_{actual}. We assume that user u_b included e_{actual} in $m_{b\to a}$ in response to an explicit or an implicit request in the message $m_{a\to b}$. Such pairs of request message and attachment $\langle m_{a\to b}, m_{b\to a}\rangle$ form the ground-truth in our evaluation framework.

Fig. 2 shows a sample mailbox M_{Beth} of user u_{Beth} containing four conversations $\{c_1, c_2, c_3, c_4\}$. During these conversations, u_{Beth} responds with an attachment twice—at time t_1 and t_2. At time t_1, in this toy example, the set of candidate items that are available in the user's mailbox for recommendation contains only the attachment from u_{Mostafa} received during the conversation c_3. At t_2, however, the set of candidates includes attachments received on all four conversation threads—from u_{Changjiu} (c_1), u_{Steve} (c_2), u_{Mostafa} (c_3), u_{Pratik} (c_3), and u_{David} (c_4)—as well as the item sent by u_{Beth} previously on the conversation thread c_1.

It is important to emphasize that our problem setting has two important constraints when recommending items, that any model should adhere to (1) a **privacy** constraint: the model can only recommend items from a user's own mailbox, and (2) a **temporal** constraint: the model can only recommend items that are already present in the user's mailbox at the time of recommendation.

3.1 Attachment retrieval

In addition to the above domain-specific constraints, we limit our setup to using a standard IR system R for retrieval, and cast the problem that the model needs to solve as a query formulation task.

Using an existing IR system has the practical benefit that one only needs to maintain a single system in contrast to the alternative where a separate attachment recommendation engine needs to be maintained. The model is presented with a message m_{req} containing an explicit or an implicit content request. The model is tasked with generating a query that can be submitted to the retrieval system R that retrieves a set of ranked messages M_R from the user's mailbox. Under this assumption, the retrieval system R is treated as a black box, and we are only interested in optimizing the query formulation model. Note that a query is only formulated when it is clear that an item needs to be attached to a reply message (§5.1), such as is the topic of [16, 17].

To extract a ranked list of attachable items from search engine R, we adopt an approach popular in entity retrieval frameworks [5] where an entity model is the mixture of document models that the entity is associated with. For a given query q issued at time t' in the mailbox of user u, attachable items $e \in E$ are then ranked in decreasing order of

$$P\left(e \mid q, u, t'\right) \propto \frac{1}{Z_1\left(e, u, t'\right)} \sum_{\substack{m \in M \\ t_m < t'}} S_R\left(m \mid q\right) f\left(e \mid m\right) \qquad (1)$$

where $S_R\left(m \mid q\right)$ is the relevance score for message m given query q according to retrieval system R, t_m is the timestamp when the message m appeared first in the mailbox M, t' is time when the model needs to make the recommendation, $f\left(e \mid m\right)$ denotes the association strength between message m and item e, and $Z_1\left(e, u, t'\right) = \sum_{\substack{m \in M \\ t_m < t'}} f\left(e \mid m\right)$ is a normalization constant. The normalization constant $Z_1\left(e, u, t'\right)$ avoids a bias towards attachable items that are associated with many messages (e.g., electronic business cards).

We associate messages with an attachable item according to the presence of the item within a message and its surrounding messages within the conversation: $f\left(e \mid m\right) = \mathbb{1}_{\text{context}(m)}\left(e\right)$. In this paper, we take context(m) to be all messages m' in the same conversation c_m as message m that occurred before the time of recommendation, i.e., $t_{m'} < t'$. Note that the exact definition of an attachable item depends on the email domain and can include individual files, file bundles and hyperlinks to external documents amongst others (see §5.3).

3.2 Evaluating query formulations

Once we have extracted $\langle m_{\text{req}}, e_{actual} \rangle$ pairs from an email corpus, each request message m_{req} is presented to the query formulation model that we want to evaluate. The model generates a query q conditioned on the message m_{req}. The query q is submitted to retrieval system R and attachable items extracted from the retrieved messages are determined according to Eq. 1. Given the ranked list of attachable items $E_{\text{retrieved}}$ and the expected item e_{actual} we can compute standard rank-based IR metrics such as MRR and NDCG (§5.5). We report the mean metric over all $\langle m_{\text{req}}, e_{actual} \rangle$ pairs extracted from the corpus.

Our approach of using the historical information from an email corpus for evaluation is comparable to the application of *click-through* data for similar purposes in Web search. In the document retrieval scenario, a user's click on a document d on the search result page is considered an implicit vote of confidence on the relevance of d to the query. Learning to rank models can be trained on this *click-through* data [27, 35, 62] if human relevance judgments are not available in adequate quantity. By explicitly attaching a file e_{actual}, similarly, the user of an email system provides a strong indication that recommending e_{actual} at the time of composing m_{res} would have been useful. We can use this information to train and evaluate supervised models for ranking attachments at the time of email composition.

4 QUERY FORMULATION MODEL

We first introduce a method for generating pseudo training data [3, 9] without the need for manual assessments. Silver-standard queries are algorithmically synthesized for each request/attachment pair and consequently scored by measuring the query's ability to retrieve the relevant attachment (§4.1). The request/query pairs part of the pseudo training collection are then used to train a convolutional neural network (§4.2) that learns to extract query terms from a request message. In this paper, we use a convolutional architecture rather than a recurrent one, as we intend to model term importance by term context without relying on the exact ordering of terms.

4.1 Model training

The candidate silver queries are extracted for request-response pairs $\langle m_{\text{req}}, m_{\text{res}} \rangle$ in a training set. Given a request message m_{req} and its associated target attachable item $e_{actual} \in E_{m_{\text{res}}}$ that is attached to reply m_{res}, where $E_{m_{\text{res}}}$ is the set of items attached to m_{res}, the objective is to select the k terms that are most likely to retrieve item e_{actual} according to Eq. 1.

In the ideal case, one considers the powerset of all terms within request message m_{req} as candidate silver queries [31, 65]. However, considering all terms is computationally intractable in our case as email messages tend to average between 70 to 110 tokens (Table 2).

In order to circumvent the intractability accompanied with computing the powerset of all terms in a message, we use the following stochastic strategy to select a fixed number of candidate query terms that we compute the powerset of. We consider two sources of query terms. The first source of candidate query terms consists of **subject** terms: topic terms in the subject of the request message. Email subjects convey relevance and context [60] and can be seen as a topical summary of the message. For the second source of query terms, we consider **recallable** terms: infrequent terms that occur frequently in messages $M_{\text{recallable}}\left(e_{actual}, t'\right) = \{m \in M \mid t_m < t', e_{actual} \in E_m\}$ that contained item e_{actual} and occur at least once in the request message. That is, we gather all terms that have the potential to retrieve e_{actual} (according to Eq. 1) and select those terms that occur in at least 30% of messages $M_{\text{recallable}}\left(e_{actual}, t'\right)$ and occur in less than 1% of all messages.

To construct candidate silver queries for a request message m_{req}, we follow the strategy as outlined in Algorithm 1 that mimics the boolean query formulation process of Salton et al. [51]. Candidate terms are selected from either the **subject** or **recallable** source in increasing order of document frequency (i.e., infrequent terms first). Unwanted terms, such as stopwords, digits, punctuation and the names of the sender and recipients that occur in the email headers, are removed. Afterwards, we take the candidate queries $\tilde{Q}_{m_{\text{req}}}$ to be all possible subsets of candidate terms (excluding the empty set).

Once we have obtained the set of candidate queries $\tilde{Q}_{m_{\text{req}}}$ for request message m_{req} we score the candidate queries as follows. For every $\tilde{q} \in \tilde{Q}_{m_{\text{req}}}$ we rank email messages using retrieval system R according to $\tilde{q}$. We then apply Eq. 1 to obtain a ranking over items $E_{u,t'}$ in the mailbox of user u at time t'. As we know the target item e_{actual} to be retrieved for request message m_{req}, we

Algorithm 1: Candidate query terms are selected by choosing a random query term source and selecting the query term with the lowest document frequency while ignoring unwanted query terms [51]. The `isUnwanted` predicate is true when the term is a stopword, contains a digit, contains punctuation or equals the names of the email sender/recipients. After selecting k terms, we consider the powerset of selected terms as silver queries.

Data: request message m_{req}, query term budget k

Result: candidate silver queries $\tilde{Q}_{m_{\text{req}}}$

set of candidate terms $T \leftarrow \{\}$;

while (*term candidates left* $\wedge$ $|T| < k$) **do**

  $S \leftarrow$ uniformly random choose **subject** or **recallable terms**;

  if $S = \varnothing$ **then**

    continue

  $t \leftarrow \arg\min_{t \in S} \text{df}(t)$;

  if $\neg$`isUnwanted`(t) **then**

    $T = T \cup \{t\}$;

  $S \leftarrow S \setminus \{t\}$

$\tilde{Q}_{m_{\text{req}}} \leftarrow 2^T - \{\varnothing\}$

quantify the performance of candidate query $\tilde{q}$ by its reciprocal rank, $\text{score}(\tilde{q}) = \frac{1}{\text{rank}(e_{actual})} \in (0, 1]$ where $\text{rank}(e_{actual})$ denotes the position of item e_{actual} (Eq. 1) in the item ranking.

After computing the score for every candidate silver query, we group queries that perform at the same level (i.e., that have the same score) for a particular request message m_{req}. We then apply two post-processing steps that improve silver-standard query quality based on the trade-off between query broadness and specificness. Following Salton et al. [51] on boolean query formulation, specific queries are preferred over broad queries to avoid loss in precision. Queries can be made more specific by adding terms. Consequently, within every group of equally-performing queries, we remove subset queries whose union results in another query that performs at the same level as the subsets. For example, if the queries *"barack obama"*, *"obama family"* and *"barack obama family"* all achieve the same reciprocal rank, then we only consider the latter three-term query and discard the two shorter, broader queries. An additional argument for the strategy above follows from the observation that any term not part of the query is considered as undesirable during learning. Therefore, including all queries listed above as training material would introduce a negative bias against the terms *"barack"* and *"family"*. However, queries that are too specific can reduce the result set [51] or cause query drift [40]. Therefore, the second post-processing step constitutes the removal of supersets of queries that perform equal or worse. The intuition behind this is that the inclusion of the additional terms in the superset query did not improve retrieval performance. For example, if queries *"barack obama"* and *"barack obama president"* perform equally well, then the addition of the term *"president"* had no positive impact on retrieval. Consequently, including the superset query (i.e., *"barack obama president"*) in the training set is likely to motivate the inclusion of superfluous terms that negatively impact retrieval effectiveness.

4.2 A convolutional neural network for ranking query terms

After obtaining a set of candidate queries $\tilde{Q}_{m_{\text{req}}}$ for every request/item pair $\langle m_{\text{req}}, e_{actual} \rangle$ in the training set, we learn to select query terms from email threads using a convolutional neural network model that convolves over the terms contained in the email thread.

Rank	Term	Score
1.	initech	0.20
2.	initech	0.18
3.	transition	0.15
4.	EoR	0.10
5.	david	0.07
6.	...	

Figure 3: Terms in the request message of Fig. 1 are ranked by our model. In addition to the terms, the model also ranks a (EoR) token that specifies the query end. The final query becomes *"initech transition"* as duplicate terms are ignored.

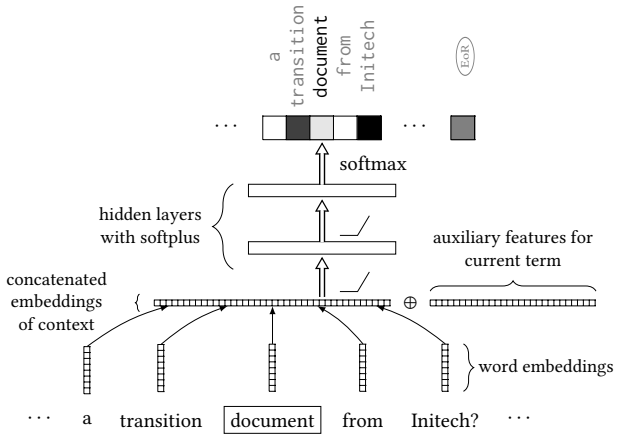


Figure 4: The model convolves over the terms in the message. For every term, we represent it using the word representations of its context (learned as part of the model). After creating a representation of the term's context, we concatenate auxiliary features (Table 1). At the output layer, a score is returned as output for every term in the message. The softmax function converts the raw scores to a distribution over the message terms and the (EoR) token. Grayscale intensity in the distribution depicts probability mass.

Every term is characterized by its context and term importance features that have been used to formulate queries in previous work [6, 14, 24, 31, 37, 63, 68]. Our model jointly learns to (1) generate a ranking of message terms, and (2) determine how many terms of the message term ranking should be included in the query. In order to determine the number of terms included in the query, the model learns to rank an end-of-ranking token (EoR) in addition to the message terms. Fig. 3 shows a term ranking for the example in Fig. 1. Terms in the request message are ranked in decreasing order of the score predicted by our model. Terms appearing at a lower rank than the (EoR) are not included in the query.

Our convolutional neural network (CNN) term ranking model is organized as follows; see Fig. 4 for an overview. Given request message m_{req}, we perform a convolution over the n message terms $w_1, \ldots, w_n$. Every term w_k is characterized by (1) the term w_k itself, (2) the $2 \cdot L$ terms, $w_{k-L}, \ldots, w_{k-1}, w_{k+1}, \ldots, w_{k+L}$, surrounding term w_k where L is a context width hyperparameter, and (3) auxiliary query term quality features (see Table 1). For every term in the message, the local context features (1st part of Table 1) are looked up in term embedding matrix W_{repr} (learned as part of the model) and the auxiliary features (part 2-4 of Table 1) are computed. For the auxiliary features, we apply min-max feature scaling on

Table 1: Overview of term representation (learned as part of the model) and auxiliary features.

Context features (learned representations)	
`term`	Representation of the term.
`context`	Representations of the context surrounding the term.
Part-of-Speech features	
`is_noun`	POS tagged as a noun [6]
`is_verb`	POS tagged as a verb
`is_other`	POS tagged as neither a noun or a verb
Message features	
`is_subject`	Term occurrence is part of the subject [14]
`is_body`	Term occurrence is part of the body [14]
`Abs. TF`	Abs. term freq. within the message [63]
`Rel. TF`	Rel. term freq. within the message [63]
`Rel. pos.`	Rel. position of the term within the message
`is_oov_repr`	Term does not have a learned representation
Collection statistics features	
`IDF`	Inverse document frequency of the term [63]
`TF-IDF`	`TF` × `IDF` [63]
`Abs. CF`	Abs. collection freq. within the collection
`Rel. CF`	Rel. collection freq. within the collection
`Rel. Entropy`	KL divergence from the unsmoothed collection term distribution to the smoothed ($\lambda = 0.5$) document term distribution [37]
`SCQ`	Similarity Collection/Query [68]
`ICTF`	Inverse Collection Term Frequency [31]
`Pointwise SCS`	Pointwise Simplified Clarity Score [24]

the message-level such that they fall between 0 and 1. The concatenated embeddings and auxiliary feature vector, are fed to the neural network. At the output layer, the network predicts a term ranking score, $g\left(w_k, m_{\text{req}}\right)$, for every term. In addition, a score for the (EoR) token, $h\left(m_{\text{req}}\right)$, is predicted as well. The (EoR) score function h takes the same form as the term score function g, but has a separate set of parameters (due to its input features having a different distribution) and takes as input an aggregated vector that represents the whole message. More specifically, the input to the (EoR) score function is the average of the term representations and their auxiliary features. The ranking scores are then transformed into a distribution over message terms and the (EoR) token as follows:

$$P\left(w_k \mid m_{\text{req}}\right) = \frac{1}{Z_2} e^{g(w_k, m_{\text{req}})} \quad \text{and} \quad P\left(\text{(EoR)} \mid m_{\text{req}}\right) = \frac{1}{Z_2} e^{h(m_{\text{req}})}$$

with $Z_2\left(m_{\text{req}}\right) = e^{h(m_{\text{req}})} + \sum_{l=0}^{|m_{\text{req}}|} e^{g(w_l, m_{\text{req}})}$ as a normalization constant that normalizes the raw term scores such that we obtain a distribution over message terms. For every query $\tilde{q} \in \tilde{Q}$, the ground-truth distribution equals:

$$Q\left(w_k \mid \tilde{q}\right) = \alpha \cdot \frac{\mathbb{1}_{\tilde{q}}\left(w_k\right)}{\#\left(w_k, m_{\text{req}}\right) \cdot |\tilde{q}|} \quad \text{and} \quad Q\left(\text{(EoR)} \mid \tilde{q}\right) = (1-\alpha) \tag{2}$$

where $\alpha = 0.95$ is a hyperparameter that determines the probability mass assigned to the (EoR) token and $\mathbb{1}_{\tilde{q}}\left(w_k\right)$ is the indicator function that evaluates to 1 when term w_k is part of silver query $\tilde{q}$. The frequency count $\#\left(w_k, m_{\text{req}}\right)$ denotes the number of times term w_k occurs in message m_{req} and is included such that frequent and infrequent message terms are equally important. Eq. 2 assigns an equal probability to every unique term in message m_{req} that occurs in silver query $\tilde{q}$. Our cost function consists of two objectives. The first objective aims to make $P\left(\cdot \mid m_{\text{req}}\right)$ close to $Q\left(\cdot \mid \tilde{q}\right)$ by minimizing the cross entropy:

$$L_{\text{xent}}\left(\theta \mid m_{\text{req}}, \tilde{q}\right) = -\sum_{\omega \in \Omega} Q\left(\omega \mid \tilde{q}\right) \log\left(P\left(\omega \mid m_{\text{req}}\right)\right) \tag{3}$$

where $\Omega = \left(w_1, \ldots, w_n, \text{(EoR)}\right)$ is the sequence of all terms in the message m_{req} concatenated with the end-of-ranking token. Eq. 3 promotes term ranking precision as it causes terms in the silver query to be ranked highly, immediately followed by the end-of-ranking token. The second objective encourages term ranking recall by dictating that the (EoR) token should occur at the same rank as the lowest-ranked silver query term:

$$L_{\text{cutoff}}\left(\theta \mid m_{\text{req}}, \tilde{q}\right) = \left(\min_{w \in m_{\text{req}}}\left(P\left(w \mid m_{\text{req}}\right)\right) - h\left(m_{\text{req}}\right)\right)^2 \tag{4}$$

The two objectives (Eq. 3-4) are then combined in a batch objective:

$$L\left(\theta \mid B\right) = \frac{1}{|B|} \sum_{(m,\tilde{q}) \in B} \text{score}\left(\tilde{q}\right) \Big(L_{\text{xent}}\left(\theta \mid m, \tilde{q}\right) + L_{\text{cutoff}}\left(\theta \mid m, \tilde{q}\right) \Big) + \frac{1}{2\lambda} \sum_{W \in \theta_W} \sum_{ij} W_{ij}^2 \tag{5}$$

where B is a uniformly random sampled batch of message/query pairs, θ_W is the set of parameter matrices and λ is a weight regularization parameter. Objective 3 resembles a list-wise learning to rank method [11] where a softmax over the top-ranked items is used. Eq. 5 is then optimized using gradient descent.

5 EXPERIMENTAL SET-UP

5.1 Research questions

Having described our proactive email attachment recommendation framework and query formulation model, we now detail the three research question that we seek to answer.

RQ1 Do convolutional neural networks (CNN) improve ranking efficacy over state-of-the-art query formulation methods?

What if we consider the different fields (subject and body) in the email message when selecting query terms? To what extent do methods based on selecting the top ranked terms according to term scoring methods (e.g., TF-IDF, RE) perform? Can CNNs outperform state-of-the-art learning to rank methods? What can we say about the length of the queries extracted by the different methods?

RQ2 When do CNNs work better than non-neural methods on the attachable item recommendation task?

In the case that CNNs improve retrieval effectiveness over query extraction methods: what can we say about the errors made by CNNs? In particular, in what cases do our deep convolutional neural networks perform better or worse compared to the query term ranking methods under comparison?

RQ3 What features are most important when training CNNs?

Are all types of features useful? Can we make any inferences about the email domain or the attachable item recommendation task?

5.2 Experimental design

We operate under the assumption that an incoming message has been identified as a request for content. A query is then formulated from the message using one of the query formulation methods (§5.4). To answer the research questions posed in §5.1, we compare CNNs with existing state-of-the-art query term selection methods on enterprise email collections (**RQ1**). In addition, we look at the

Table 2: Overview of the enterprise email collections used in this paper: Avocado (public) and PIE (proprietary).

	Avocado	PIE
Messages	928,992	1,047,311
Message length (terms)	112.33 ± 244.01	74.70 ± 551.88
Threads	804,010	381,448
Thread lengths	1.19 ± 0.70	2.75 ± 3.65
Time period	3 years, 8 months	1 year
Attachable entities	50,462	28,725
Impressions per item	3.48 ± 2.55	2.79 ± 1.36
Messages with an item	311,478	152,649
no thread history	288,099	69,796
all items filtered (§5.3)	22,399	80,717
Request/reply pairs	980	2136
Thread history length of pairs	1.53 ± 1.13	4.04 ± 5.78
Relevant items per pair	1.22 ± 0.70	1.29 ± 1.82

query lengths generated by the formulation methods that perform best. **RQ2** is answered by examining the per-instance difference in reciprocal rank (§5.5) and a qualitative analysis where we examine the outlier examples. For **RQ3** we perform a feature ablation study where we systematically leave out a feature category (Table 1).

5.3 Data collections and pre-processing

We answer our research questions (§5.1) using two enterprise email collections that each constitute a single tenant (i.e., an organization): (1) the Avocado collection [42] is a public data set that consists of emails taken from 279 custodians of a defunct information technology company, and (2) the Proprietary Internal Emails (PIE) collection is a proprietary dataset of Microsoft internal emails obtained through an employee participation program. We perform cross-validation on the collection level. That is, when testing on one collection, models are trained and hyperparameters are selected on the other collection (i.e., train/validate on Avocado, test on PIE and vice versa). Models should generalize over multiple tenants (i.e., organizations) as maintaining specialized models is cumbersome. In addition, model effectiveness should remain constant over time to avoid frequent model retraining. Consequently, topical regularities contained within a tenant should not influence our comparison. Furthermore, privacy concerns may dictate that training and test tenants are different. On the training set, we create a temporal 95/5 split for training and validation/model selection. On the test set, all instances are used for testing only. Attachable items consist of file attachments and URLs; see Table 2.

The training and test instances are extracted, for every collection independently, in the following unsupervised manner. File attachments and normalized URLs are extracted from all messages. We remove outlier items by trimming the bottom and top 5% of the attachable item frequency distribution. Infrequent items are non-retrievable and are removed in accordance to our experimental design (§5.2). However, in this paper we are interested in measuring the performance on retrieving attachable items that are in the "torso" of the distribution and, consequently, frequent items (e.g., electronic business cards) are removed as well. Any message that links to an attachable item (i.e., URL or attachment) and the message preceding it is considered a request/reply instance. In addition, we filter request/reply instances containing attachable items that (a) occurred previously in the same thread, or (b) contain attachable items that do not occur in the user's mailbox before the time of the request message (see §5.2). Mailboxes are indexed and searched using Indri [57, 58]. For message retrieval, we use the Query-Likelihood Model (QLM) with Dirichlet smoothing [66] where the smoothing parameter (μ) is set to the average message length [5, 59]. At test time, query formulation methods extract query terms from the request message, queries are executed using the email search engine of the user (i.e., Indri) and attachable items are ranked according to Eq. 1. Rankings are truncated such that they only contain the top-1000 messages and top-100 attachable items. The ground truth consists of binary labels where items part of the reply message are relevant.

5.4 Methods under comparison

As the attachable item recommendation task is first introduced in this paper, there exist no methods directly aimed at solving this task. However, as mentioned in the related work section (§2), there are two areas (prior art search and verbose query reduction) that focus on extracting queries from texts. Consequently, we use computationally tractable methods (§2) from these areas for comparison: (1) single features, i.e., term frequency (TF), TF-IDF, logTF-IDF, relative entropy (RE), used for prior art retrieval [14, 37, 63, 64] where the top-k unique terms are selected from either the subject, the body or both. Hyperparameters $1 \leq k \leq 15$ and, in the case of RE, $\lambda = 0.1, \ldots, 0.9$ are optimized on the validation set, (2) the learning to rank method for query term ranking proposed by Lee et al. [32] for the verbose query reduction task. To adapt this method for our purposes, we use the domain-specific features listed in Table 1 (where the representations are obtained by training a Skip-Gram word2vec model with default parameters on the email collection), only consider single-term groups (as higher order term groups are computationally impractical during inference) and use a more powerful pairwise RankSVM [27] (with default parameters [53]) instead of a pointwise approach. Feature value min-max normalization is performed on the instance-level. The context window width $L = 3, 5, \ldots, 15$ is optimized on the validation set. In addition, we consider the following baselines: (3) all terms (Full) are selected from either the subject, the body or both, (4) random terms, selected from the subject, the body or both, where we either select k unique terms randomly (Random k) or a random percentage p of terms (Random %). Hyperparameters $1 \leq k \leq 15$ and $p = 10\%, 20\%, \ldots, 50\%$ are optimized on the validation set. Finally, we consider a pointwise alternative to the CNN model: (5) CNN-p with the logistic function at the output layer (instead of the softmax) and terms are selected if their score exceeds a threshold optimized on the validation set F1 score (instead of the (EoR) token).

The CNN models are trained for 30 iterations using Adam [30] with $\alpha = 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$; we select the iteration with lowest data loss on the validation set. Word embeddings are 128-dim., the two hidden layers have 512 hidden units each, with dropout ($p = 0.50$) and the softplus function. Weights are initialized according to [18]. Batch size $|B| = 128$ and regularization lambda $\lambda = 0.1$. The context window width $L = 3, 5, \ldots, 15$ is optimized on the validation set. For word embeddings (both as part of the CNN and RankSVM), we consider the top-60k terms. Infrequent terms share a representation for the unknown token.

5.5 Evaluation measures

To answer **RQ1**, we report the Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG) and Precision@5 (P@5) measures.[2] For **RQ2**, we examine the pairwise differences in

[2]Computed using `trec_eval`: https://github.com/usnistgov/trec_eval

Table 3: Comparison of CNN with state-of-the-art query formulation methods (§5.4) on the Avocado and PIE collections. The numbers reported on Avocado were obtained using models trained/validated on PIE and vice versa (§5.3). Significance is determined using a paired two-tailed Student t-test (* $p < 0.10$; ** $p < 0.05$) [55] between CNN and the second best performing method (in italic).

	Avocado			PIE		
	MRR	NDCG	P@5	MRR	NDCG	P@5
Full field, single features and random (subject)						
Full	0.2286	0.3097	0.0686	0.3338	0.4621	0.1088
TF	0.2280	0.3095	0.0686	0.3315	0.4600	0.1079
TF-IDF	0.2250	0.3073	0.0704	0.3390	0.4663	0.1090
logTF-IDF	0.2280	0.3095	0.0686	0.3315	0.4600	0.1079
RE	0.2223	0.3038	0.0698	*0.3391*	*0.4664*	*0.1095*
Random *k*	0.2143	0.2932	0.0647	0.3266	0.4553	0.1063
Random %	0.1481	0.2104	0.0467	0.2749	0.4013	0.0889
Full field, single features and random (body)						
Full	0.1248	0.1930	0.0377	0.2115	0.3376	0.0672
TF	0.1025	0.1719	0.0309	0.2094	0.3358	0.0660
TF-IDF	0.1507	0.2213	0.0459	0.2237	0.3481	0.0722
logTF-IDF	0.1109	0.1755	0.0311	0.1914	0.3180	0.0627
RE	0.1441	0.2128	0.0424	0.2198	0.3430	0.0699
Random *k*	0.0785	0.1394	0.0229	0.1781	0.3078	0.0568
Random %	0.1030	0.1646	0.0325	0.1887	0.3128	0.0606
Full field, single features and random (subject + body)						
Full	0.1995	0.2785	0.0612	0.3087	0.4406	0.0972
TF	0.1783	0.2653	0.0551	0.3005	0.4334	0.0953
TF-IDF	0.2097	0.2933	0.0649	0.3100	0.4397	0.0991
logTF-IDF	0.1858	0.2726	0.0592	0.2747	0.4098	0.0871
RE	0.2138	0.2980	0.0649	0.3200	0.4489	0.1023
Random *k*	0.1404	0.2148	0.0436	0.2721	0.4076	0.0886
Random %	0.1753	0.2514	0.0520	0.2592	0.3941	0.0822
Learning-to-rank methods (subject + body)						
RankSVM	0.1650	0.2425	0.0497	0.3079	0.4392	0.0980
CNN-p	*0.2319*	*0.3129*	*0.0708*	0.3347	0.4630	0.1087
CNN	**0.2455***	**0.3313****	**0.0770****	**0.3492****	**0.4744****	**0.1123**

terms of Reciprocal Rank (RR). In the case of **RQ3**, we measure the relative difference in MRR when removing a feature category.

6 RESULTS & DISCUSSION

6.1 Overview of experimental results

RQ1 Table 3 shows the recommendation results of attachable items in enterprise email collections (§5.3).

We see that CNN outperforms all other query formulation methods on both enterprise email collections. Significance is achieved (MRR) between CNN and the second-best performing methods: CNN-p and RE (subject), respectively, on the Avocado and PIE. The methods that select terms only from the email subject perform strongly on both collections. Within the set of subject methods (1st part of Table 3), we also observe that there is little difference between the methods. In fact, for Avocado, simply taking the subject as query performs better than any of the remaining subject-based methods. Subjects convey relevance and context [60] and can compactly describe the topic of a content request. However, in order to generate better queries, we need to extract additional terms from the email body as email subjects tend to be short.

The same methods that we used to extract terms from the subject perform poorly when only presented with the body of the email (2nd part of Table 3). When allowing the methods to select terms from the full email (subject and body), we see a small increase in retrieval performance (3rd part of Table 3) compared to body-only terms. However, none of the methods operating on the full email message manage to outperform the subject by itself.

Our learning to rank (LTR) query term methods (last part of Table 3) outperform the subject-based methods (ignoring RankSVM). This comes as little surprise, as the presence of a term in the subject is incorporated as a feature in our models (Table 1). The reason why RankSVM, originally introduced for reducing long search queries, performs poorly is due to the fact that its training procedure fails to deal with the long length of emails. That is, RankSVM is trained by measuring the decrease in retrieval effectiveness that occurs from leaving one term out of the full email message (i.e., top-down). This is an error-prone way to score query terms as emails tend to be relatively long (Table 2). Conversely, our approach to generate silver query training data (§4.1) considers groups of query terms and the reference query is constructed in a bottom-up fashion.

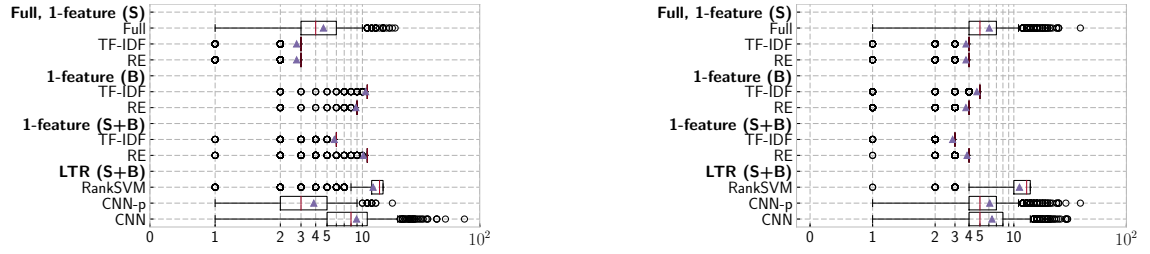
Fig. 5 shows the distribution of generated query lengths for the most prominent methods (TF-IDF, RE, RankSVM and the neural networks). On both collections, subject-oriented methods (TF-IDF, RE) seem to extract queries of nearly all the same length. When considering the full subject field, we see that its length varies greatly with outliers of up to 70 query terms. Methods that extract terms from the email body or the full email generate slightly longer queries than the TF-IDF and RE subject methods. The methods that learn to rank query terms generate the longest queries. While RankSVM selects query terms according to a global rank cut-off, the CNNs select a variable number of terms for every request message. Consequently, it comes as no surprise that we observe high variance within the CNN-generated query lengths. In addition, we observe that CNNs have a similar query length distribution as the subject queries. This is due to the fact that the CNNs actually expand the subject terms, as we will see in the next section.

6.2 Analysis of differences

RQ2 Fig. 6 shows the per-instance differences between CNN and the full email subject as query.

For about 60% of request messages both query generation methods (full subject and CNN) generate queries that perform equally good. In fact, MRR on this subset of instances is 9% (Avocado) and 17% (PIE) better than the best performance over the full test set (Table 3). Do both methods generate identical queries on this subset? The average Jaccard similarity between the queries extracted by both methods is 0.55 (Avocado) and 0.62 (PIE). This indicates that, while query terms extracted by either method overlap, there is a difference in query terms that does not impact retrieval. Upon examining the differences we find that, for the subject, these terms are stopwords and email subject abbreviations (e.g., RE indicating a reply). In the case of CNN, the difference comes from email body terms that further clarify the request. We find that the CNN builds upon the subject query (excluding stopwords) using terms of the body. However, for the 60% of request instances with no difference in performance (Fig. 6), the subject suffices to describe the request.

What can we say about the remaining 40% of queries where there is an observable difference? A closer look at Fig. 6 shows that there are extreme peaks at both sides of the graph and that neither method fully dominates the other. Upon examining the outliers, we find the following trends: (1) When the subject is indescriptive of the email content (e.g., the subject is *"important issue"*) then the CNN can extract better terms from the email body. (2) Topic drift within conversations negatively impacts the retrieval effectiveness



(a) Avocado (b) PIE

Figure 5: Query length distribution according to the most prominent methods (S and B denote subject and body fields, resp.). TF-IDF, RE and RankSVM select a fixed number of terms for all queries, whereas CNNs select a variable number of terms.

Figure 6: Per-instance differences in Reciprocal Rank (RR) between the email subject query and the CNN query on Avocado. The plot for PIE is qualitatively similar. Positive bars (left) indicate instances where the subject performs better than CNN and vice versa for negative (right).

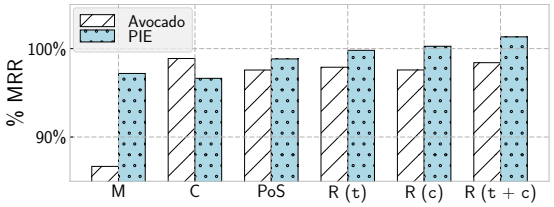


Figure 7: Feature ablation study for the CNN model on the Avocado (stripes) and PIE (dots) benchmarks. One of the following feature categories (Table 1) is systematically left out: Message (M), Collection statistics (C), Part-of-Speech (PoS), `term` (`t`), `context` (`c`) or all representations (`t` + `c`).

of the subject as a query. However, long threads do not necessarily exhibit topic drift as in some cases the subject remains a topical representation of the conversation. (3) Mentions of named entities constitute effective query terms. For example, the name of a person who is responsible for an attachable item tends to improve retrieval effectiveness over using the subject as a query, (4) Long queries generated by the CNN can disambiguate a request and perform much better than the subject query. (5) In most cases where the subject query outperforms the CNN, this is due to the fact that the CNN model extracts too many noisy terms and creates query drift.

6.3 Feature importance

RQ3 Fig. 7 depicts a feature ablation study where we systematically leave out one of the feature categories.

We observe that both local (message and part-of-speech) and global (collection) features are of importance. When comparing the behavior of the enterprise email collections (§5.3), we see that the message (M) features have a significant ($p < 0.10$) impact on both collections. The collection statistics (C) yield a significant ($p < 0.10$) difference in the case of PIE; no significant differences were observed in the remaining cases. In addition, while Avocado benefits from the learned representations, the inclusion of the representations of the context slightly decreases performance on PIE. This can be explained by our evaluation setup (§5.2) where models evaluated on PIE are trained using Avocado and vice versa. Therefore, it is likely that the model learns certain patterns present from the data-scarce Avocado collection (Table 2) that causes false positive terms to be selected for PIE.

7 CONCLUSIONS

We introduced a novel proactive retrieval task for recommending email attachments that involves formulating a query from an email request message. An evaluation framework was proposed that extracts labeled request/attachment instances from an email corpus containing request/reply pairs automatically. Candidate queries, which we refer to as *silver queries*, are synthesized for request/attachment instances and a deep convolutional neural network (CNN) is trained using the silver queries that learns to extract query terms from request messages. We find that our framework extracts instances that are usable for training and testing. Our CNN, which we train using silver queries, significantly outperforms existing methods for extracting query terms from verbose texts. Terms occurring in the subject of the email are representative of the request and formulating a query using the subject is a strong baseline. A study of the per-instance MRR differences show that the CNN and subject query perform quite differently for 40% of instances. A qualitative analysis suggests that our CNN outperforms the subject query in cases where the subject is indescriptive. In addition, mentions of named entities constitute good query terms and lengthy queries disambiguate the request. In cases when the subject query outperforms the CNN, it is due to noisy terms being selected from the email body. A feature ablation study shows that both local (i.e., message) and global (i.e., collection) features are important.

Our work has the following limitations. (1) In this paper we only consider terms occurring in the request message as candidates. While this prevents the term candidate set from becoming too large, it does limit the ability for methods to formulate expressive queries in the case where request messages are concise. (2) The retrieval model used in this paper, a language model with Dirichlet smoothing, is ubiquitous in retrieval systems. However, smoothing allows the search engine to deal with verbose queries [66] that contain terms absent from the messages. Subjects often contain superfluous terms (e.g., email clients prepend *FW* to the subjects of forwarded messages). Consequently, our findings may change when considering other retrieval model classes, such as boolean models or semantic matching models. Future work includes analysis of the effect of various pre-processing steps (e.g., signature removal,

thread merging, phrase-level tokenization), a study of the effect of training/evaluating query formulation models on the email of a single organization to measure the effect of organization-specific properties and the incorporation of the social graph in the email domain. In addition, structured queries with operators searching different fields (e.g., recipients, subject) can improve performance. Finally, we assumed a single model for all mailboxes; per-mailbox personalized models are likely to generate better queries.

Acknowledgments. The authors would like to thank Maarten de Rijke and the anonymous reviewers for their valuable comments. The first author was supported, before and after his employment at Microsoft, by the Bloomberg Research Grant program and the Google Faculty Research Award scheme. All content represents the authors' opinions, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Q. Ai, S. T. Dumais, N. Craswell, and D. Liebling. Characterizing email search using large-scale behavioral logs and surveys. In *WWW*, p. 1511–1520, 2017.
[2] J. Allan, B. Croft, A. Moffat, and M. Sanderson. Frontiers, challenges, and opportunities for information retrieval. In *SIGIR Forum*, volume 46, p. 2–32. ACM, 2012.
[3] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *SIGIR*, p. 1073–1082. ACM, 2011.
[4] N. Balasubramanian, G. Kumaran, and V. R. Carvalho. Exploring reductions for long web queries. In *SIGIR*, p. 571–578. ACM, 2010.
[5] K. Balog, L. Azzopardi, and M. De Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, p. 43–50. ACM, 2006.
[6] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *SIGIR*, p. 491–498. ACM, 2008.
[7] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log. In *Workshop on Web Search Click Data*, p. 8–14. ACM, 2009.
[8] J. R. Benetka, K. Balog, and K. Nørvåg. Anticipating information needs based on check-in activity. In *WSDM*, p. 41–50. ACM, 2017.
[9] R. Berendsen, M. Tsagkias, W. Weerkamp, and M. De Rijke. Pseudo test collections for training and tuning microblog rankers. In *SIGIR*, p. 53–62. ACM, 2013.
[10] J. Budzik and K. Hammond. Watson: Anticipating and contextualizing information needs. In *ASIS*, volume 36, p. 727–740. Information Today, 1999.
[11] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML*, p. 129–136. ACM, 2007.
[12] V. R. Carvalho and W. W. Cohen. On the collective classification of email speech acts. In *SIGIR*, p. 345–352. ACM, 2005.
[13] D. D. Castro, L. Lewin-eytan, Z. Karnin, and Y. Maarek. You've got mail, and here is what you could do with it! In *WSDM*, 2016.
[14] S. Cetintas and L. Si. Effective query generation and postprocessing strategies for prior art patent search. *JASIST*, 63(3):512–527, 2012.
[15] I. B. Crabtree, S. J. Soltysiak, and M. Thint. Adaptive personal agents. *Personal Technologies*, 2(3):141–151, 1998.
[16] M. Dredze, J. Blitzer, and F. Pereira. "Sorry, I Forgot the Attachment": Email attachment prediction. In *CEAS*, 2006.
[17] M. Dredze, T. Brooks, J. Carroll, J. Magarick, J. Blitzer, and F. Pereira. Intelligent email: reply and attachment prediction. In *IUI*, p. 321–324. ACM, 2008.
[18] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, p. 249–256, 2010.
[19] M. Golestan Far, S. Sanner, M. R. Bouadjenek, G. Ferraro, and D. Hawking. On term selection techniques for patent prior art search. In *SIGIR*, p. 803–806. ACM, 2015.
[20] D. Graus, D. van Dijk, M. Tsagkias, W. Weerkamp, and M. de Rijke. Recipient recommendation in enterprises using communication graphs and email content. In *SIGIR*, p. 1079–1082. ACM, 2014.
[21] C. Grevet, D. Choi, D. Kumar, and E. Gilbert. Overload is overloaded: email in the age of gmail. In *SIGCHI*, p. 793–802. ACM, 2014.
[22] M. Gupta, M. Bendersky, et al. Information retrieval with verbose queries. *FnTIR*, 9(3-4):209–354, 2015.
[23] P. E. Hart and J. Graham. Query-free information retrieval. *IEEE Expert*, 12(5): 32–37, 1997.
[24] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *SPIRE*, p. 43–54. Springer, 2004.
[25] E. Horvitz. Principles of mixed-initiative user interfaces. In *SIGCHI*, p. 159–166. ACM, 1999.
[26] S. Huston and W. B. Croft. Evaluating verbose query processing techniques. In *SIGIR*, p. 291–298. ACM, 2010.
[27] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, p. 133–142. ACM, 2002.
[28] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukács, M. Ganea, P. Young, et al. Smart reply: Automated response suggestion for email. In *KDD*, 2016.
[29] Y. Kim, J. Seo, and W. B. Croft. Automatic boolean query suggestion for professional search. In *SIGIR*, p. 825–834. ACM, 2011.
[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.
[31] G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *SIGIR*, p. 564–571. ACM, 2009.
[32] C.-J. Lee, R.-C. Chen, S.-H. Kao, and P.-J. Cheng. A term dependency-based approach for query terms ranking. In *CIKM*, p. 1267–1276. ACM, 2009.
[33] D. J. Liebling, P. N. Bennett, and R. W. White. Anticipatory search: using context to initiate search. In *SIGIR*, p. 1035–1036. ACM, 2012.
[34] M. Lupu, A. Hanbury, et al. Patent retrieval. *FnTIR*, 7(1):1–97, 2013.
[35] C. Macdonald and I. Ounis. Usefulness of quality click-through data for training. In *WSCD*, p. 75–79. ACM, 2009.
[36] P. P. Maglio, R. Barrett, C. S. Campbell, and T. Selker. Suitor: An attentive information system. In *IUI*, p. 169–176. ACM, 2000.
[37] P. Mahdabi, M. Keikha, S. Gerani, M. Landoni, and F. Crestani. Building queries for prior-art search. In *IRFC*, p. 3–15. Springer, 2011.
[38] K. T. Maxwell and W. B. Croft. Compact query term selection using topically related text. In *SIGIR*, p. 583–592. ACM, 2013.
[39] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. De Rijke. Learning semantic query suggestions. *The Semantic Web-ISWC*, p. 424–440, 2009.
[40] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *SIGIR*, p. 206–214. ACM, 1998.
[41] E. D. Mynatt, M. Back, R. Want, M. Baer, and J. B. Ellis. Designing audio aura. In *SIGCHI*, p. 566–573. ACM, 1998.
[42] D. Oard, W. Webber, D. Kirsch, and S. Golitsynskiy. Avocado research email collection. *Linguistic Data Consortium*, 2015.
[43] D. Odijk, E. Meij, I. Sijaranamual, and M. de Rijke. Dynamic query modeling for related content finding. In *SIGIR*, p. 33–42. ACM, 2015.
[44] K. Purcell and L. Rainie. Technology's impact on workers. Technical report, Pew Research Center, 2014.
[45] A. Qadir, M. Gamon, P. Pantel, and A. H. Awadallah. Activity modeling in email. In *NAACL-HLT*, p. 1452–1462, 2016.
[46] B. Rhodes and T. Starner. Remembrance agent: A continuously running automated information retrieval system. In *PAAMS*, p. 487–495, 1996.
[47] B. J. Rhodes. The wearable remembrance agent: A system for augmented memory. In *ISWC*, p. 123–128. IEEE, 1997.
[48] B. J. Rhodes. Margin notes: Building a contextually aware associative memory. In *IUI*, p. 219–224. ACM, 2000.
[49] B. J. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3.4):685–704, 2000.
[50] N. Ryan, J. Pascoe, and D. Morse. Enhanced reality fieldwork: the context aware archaeological assistant. In *CAA*, p. 269–274. Archaeopress, 1999.
[51] G. Salton, C. Buckley, and E. A. Fox. Automatic query formulations in information retrieval. *JASIST*, 34(4):262, 1983.
[52] N. Sawhney and C. Schmandt. Nomadic radio: speech and audio interaction for contextual messaging in nomadic environments. *TOCHI*, 7(3):353–383, 2000.
[53] D. Sculley and G. Inc. Large scale learning to rank. In *In NIPS 2009 Workshop on Advances in Ranking*, 2009.
[54] M. Shokouhi and Q. Guo. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *SIGIR*, p. 695–704. ACM, 2015.
[55] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM*, p. 623–632. ACM, 2007.
[56] Y. Song and Q. Guo. Query-less: Predicting task repetition for nextgen proactive search and recommendation engines. In *WWW*, p. 543–553. IW3C2, 2016.
[57] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *ICIA*, 2005.
[58] C. Van Gysel, E. Kanoulas, and M. de Rijke. Pyndri: a python interface to the indri search engine. In *ECIR*, volume 2017. Springer, 2017.
[59] W. Weerkamp, K. Balog, and M. De Rijke. Using contextual information to improve search in email archives. In *ECIR*, p. 400–411. Springer, 2009.
[60] S. A. Weil, D. Tinapple, and D. D. Woods. New approaches to overcoming e-mail overload. In *HFES*, volume 48, p. 547–551. SAGE, 2004.
[61] S. Whittaker and C. Sidner. Email overload: Exploring personal information management of email. In *SIGCHI*, p. 276–283. ACM, 1996.
[62] J. Xu, C. Chen, G. Xu, H. Li, and E. R. T. Abib. Improving quality of training data for learning to rank using click-through data. In *WSDM*, p. 171–180. ACM, 2010.
[63] X. Xue and W. B. Croft. Automatic query generation for patent search. In *CIKM*, p. 2037–2040. ACM, 2009.
[64] X. Xue and W. B. Croft. Transforming patents into prior-art queries. In *SIGIR*, p. 808–809. ACM, 2009.
[65] X. Xue, S. Huston, and W. B. Croft. Improving verbose queries using subset distribution. In *CIKM*, p. 1059–1068. ACM, 2010.
[66] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *TOIS*, 22(2):179–214, 2004.
[67] L. Zhao and J. Callan. Term necessity prediction. In *CIKM*, p. 259–268. ACM, 2010.
[68] Y. Zhao, F. Scholer, and Y. Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *ECIR*, p. 52–64. Springer, 2008.