Multi-view Embedding-based Synonyms for Email Search

Cheng Li[†], Mingyang Zhang[†], Michael Bendersky[†], Hongbo Deng[‡*], Donald Metzler[†], Marc Najork[†]

[†]Google, USA
[‡]Alibaba Inc., China
{chgli,mingyang,bemike}@google.com,arcatdeng@gmail.com,{metzler,najork}@google.com

ABSTRACT

Synonym expansion is a technique that adds related words to search queries, which may lead to more relevant documents being retrieved, thus improving recall. There is extensive prior work on synonym expansion for web search, however very few studies have tackled its application for email search. Synonym expansion for private corpora like emails poses several unique research challenges. First, the emails are not shared across users, which precludes us from directly employing query-document bipartite graphs, which are standard in web search synonym expansion. Second, user search queries are of personal nature, and may not be generalizable across users. Third, the size of the underlying corpora from which the synonyms may be mined is relatively small (i.e., user's private email inbox) compared to the size of the web corpus. Therefore, in this paper, we propose a solution tailored to the challenges of synonym expansion for email search. We formulate it as a multi-view learning problem, and propose a novel embedding-based model that joins information from multiple sources to obtain the optimal synonym candidates. To demonstrate the effectiveness of the proposed technique, we evaluate our model using both explicit human ratings as well as a live experiment using the Gmail Search service, one of the world's largest email search engines.

KEYWORDS

embedding; synonym expansion; personal search; email search

ACM Reference Format:
Cheng Li, Mingyang Zhang, Michael Bendersky, Hongbo Deng, Donald Metzler, 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 (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331250

1 INTRODUCTION

When using a search engine, users expect that their (usually very short) queries will be sufficient to retrieve desired and relevant information. In practice, this requires the search engine to bridge the semantic gap between the relevant document text and the query text. Query expansion, which is the process of reformulating a query to improve matching, has been proposed as an effective method for bridging this gap [24, 50]. Common techniques for query expansion include synonym expansion, word stemming, correcting spelling errors and term re-weighting.

Synonym expansion is an important query expansion method [45, 58]. Synonym expansion adds related words to search queries, aiming at bridging the semantic gap and retrieving more relevant documents. These documents would otherwise not be retrieved by the original query, and thus, synonym expansion can positively affect the document recall.

There is extensive prior work on synonym expansion for web search [20, 37, 50, 52], where abundant information can be utilized, including the entire web corpus and search logs. For example, many techniques rely on the bipartite query-document click graph to expand the query [19, 20, 52, 64]. Most recently with the success of embedding-based methods and deep learning techniques, query expansion has been further improved by considering terms that are close in the embedding space [22, 41, 54].

Compared with the large amount of prior work on synonym expansion for web search, the research on synonym expansion for email search is still at a nascent stage [41]. This is despite the fact that a large number of users are relying on email search on a daily basis to access their emails [13, 59]. It is not uncommon in the email search setting for users to type a query, hoping to retrieve an email they once saw, with no result being returned. In this situation, users exert mental effort to come up with the right set of keywords, experiment with various queries, and many of them finally give up, which ultimately leads to user dissatisfaction and frustration with email search [13].

This situation calls for more research efforts to improve the email search experience, with synonym expansion being one of the most important techniques. However, several challenges need to be resolved to enable successful application of synonym expansion to email search. First, emails are not shared across users. This results in sparsity of click information, posing difficulties for methods that rely on clicks to derive synonyms. This sparsity also makes it almost impossible to build a query-document bipartite graph, which is a standard approach in web search synonym expansion [2].

Second, user search queries are of personal nature, and may not generalize across users. Unlike web users who are more likely to use similar queries to access the public web corpus, users in email search may compose totally different queries to retrieve documents from their own corpus. For example, an email user might issue a query like *john smith schedule* to retrieve an email about John Smith's schedule, while most other users would have never issued this query. This query sparsity problem introduces many long-tail

[*] Work done while at Google.

SIGIR '19, July 21–25, 2019, Paris, France
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6172-9/19/07.
https://doi.org/10.1145/3331184.3331250

queries, making it difficult to use expansion methods that directly rely on data aggregation across queries.

Third, compared with the size of the public web corpus, the size of each individual private corpus is much smaller. This hinders mining synonyms directly from the underlying corpus [5], and further exacerbates the data sparsity problem.

Finally, due to the sensitive nature of private data, it is impossible to apply the various mining and learning techniques directly to the private queries and documents, as doing so could leak potentially sensitive information. Instead, to preserve privacy, we only use a limited number of n-grams from each query and email subject. No information from email body is ever processed. Moreover, each n-gram we use is k-anonymized, that is, it should be frequent enough to be contained in query logs of sufficiently many users [3, 21]. In addition, the maximum size n of the n-grams is set to a very small value, and only *a set of frequent n-grams* can be used without sequence information. All of these strict requirements protect users' privacy, while drastically reduce the amount of information available for learning.

As a workaround to these constraints, we might consider directly employing synonyms derived from a public external source, e.g., WordNet [58]. However, such an approach may not always be optimal for email search, which may contain specialized and idiosyncratic vocabulary. In addition, there may be a shift in semantics when moving from public to private domains. For instance, the synonyms for the term *target* in WordNet are terms like *aim*, *mark* and *prey*. However, when issuing the query *target invoice* in email search, most people intend to retrieve orders from the Target store, rather than documents that mention these WordNet synonyms.

Accordingly, to address all of the aforementioned challenges, we propose a novel multi-view embedding-based model that joins information from multiple information sources to obtain the optimal set of synonym candidates, and empirically evaluate this model using both offline data sets and live experiments via Gmail Search, one of the world's largest email search engines.

For offline evaluation, we consider a list of strong baselines, which either rely on external publicly available resources, or email search logs, including embedding based methods and classical query expansion methods that are based on click graphs.

For online experiments, in addition to comparing with the strongest baseline from offline evaluation, we include the synonyms used by Google Web Search as an additional, highly competitive, baseline [43]. This is a comprehensive list of synonyms developed for decades at Google; the synonyms were derived by various proprietary state-of-the-art algorithms that are based on user sessions, clicks, etc.

The detailed contributions of this paper are as follows.

- We propose to address the data sparsity challenge by embedding queries into different search contexts, so that the queries can be viewed from different perspectives. Specifically, we represent queries by their clicked documents, users who issued them, and neighboring queries in a search session. In this way, different views of search activity provide complementary information, alleviating the sparsity problem.
- To fully utilize the information from each view, we specifically design an embedding-based model, which is able to discover both syntactically and semantically related synonyms. We then join the information learned from each view via label-propagation based filtering and learning-to-rank.
- We empirically evaluate our models, and demonstrate that the synonyms discovered by our model significantly outperform the candidate synonyms found by strong, both publicly available as well as proprietary, baselines in both offline evaluation as well as in a live experiment using one of the world's largest email search engines.

It is also worth noting that though we focus on email search in particular, our proposed method is general enough to be applied to other personal search scenarios, including (but not limited to) cloud file storage, mobile device data and personal media search.

2 RELATED WORK

Query expansion has been extensively studied in the information retrieval literature [15, 53]. It broadens queries by adding additional terms or phrases to improve recall and precision. Synonym expansion, as an important component of query expansion, has attracted the attention of the research community. Our work is focused on embedding-based synonym expansion for email search, which can be easily applied to personal search. It is broadly related to three research fields: (a) query expansion for general search applications, (b) deep learning for query expansion, and (c) query expansion for personal search. In what follows, we provide a brief survey of these research fields.

2.1 Query expansion for general search applications

One common technique to find synonyms is to use a thesaurus, e.g., WordNet [29, 45, 58]. These curated knowledge resources can be used to find semantically similar terms to the query. Correlations between terms can be extracted from the document corpus itself as well [37, 50]. Search logs have been another important source to mine various types of information: queries in one session are likely to be related [25, 34, 39], clicked documents can be used to extract the expansion terms [20, 52], and query-click bipartite graphs can be constructed to find more terms [2, 16]. Another technique for query expansion utilizes the retrieved documents of a query for relevance feedback [19, 55] or pseudo-relevance feedback [18, 62].

2.2 Deep learning for query expansion

Deep neural models have been successfully applied to the domain of information retrieval [48, 66, 67]. Many models based on deep learning techniques have been developed for ranking based on clickthrough data [8, 35, 44, 56]. There are also a few attempts to apply embedding based approaches for query expansion. The general idea is that words with similar embeddings could be related, and synonyms can be discovered based on a similarity threshold using machine learning. In [42, 54], word2vec [47] is directly applied to the entire document corpus for query expansion. Diaz et al. [22] demonstrate that a word2vec model trained on documents related to the topic of the query achieves better performance than one trained on the global corpus. Grbovic et al. [30] proposed a context-aware embedding approach for query rewriting in sponsored search by treating a search session as a "sentence" and queries within the session as "words" along with word2vec techniques. Similar to relevance feedback, similarity functions can be learned to make queries and relevant document terms lie close together [57]. Term embeddings have also been explored to re-weight query terms [68]. Moreover, He et al. [31] proposed using sequence-to-sequence models to generate query rewrite candidates.

2.3 Query expansion for personal search

Most of the methods described above are most commonly studied in the web search setting. Recently, personal search (e.g., email search) attracted the attention of the information retrieval research community [3, 13, 59]. However, there is very little published work on query expansion in the personal search setting. Kuzi et al. [41] were the first to explore the direct application of word2vec [47] to the content of the personal mailbox of each user, together with a comparison to other classical expansion methods, e.g., probabilistic translation models and pseudo-relevance feedback. Other work on personal search employs classical expansion methods in the scenario of query auto-completion or spelling correction. In the absence of query logs, [5, 33] rely on email content to extract related terms for query auto-completion. In [14], a learning-to-rank model is trained on features from the query log of an individual user and users with high demographic similarity. Bhole et al. [6] propose a spelling correction algorithm that generates corrections directly from users' own mail data.

The novelty of this work, as compared to prior research, is that we design a comprehensive end-to-end framework that specifically addresses the challenges of data sparsity and anonymity in email search, and that is able to learn from user interactions. We formulate our framework in a multi-view learning setting, where embeddings from multiple user interactions (click, query session, user distribution) are combined in a unified model.

3 EMBEDDING-BASED SYNONYM EXPANSION

In this section, we will present embedding-based synonym expansion for email search. Our approach has three steps: learning from multiple views, label propagation filtering, and candidate ranking. The general framework of the approach is shown in Figure 1. Since our main focus in this paper is on the email search scenario, we shall use the terms *email* and *document* interchangeably. However, it is important to note that the approach discussed here can be easily generalized to other personal search scenarios.

Note that the privacy constraints of the email search scenario set two important limitations on the proposed methods. First, we cannot build personalized models for each user. Therefore, we only consider global models that are applicable *across users* in this paper. Second, our methods operate over sets of k-anonymized n-grams, rather than complete text sequences, which precludes us from using sequence learning techniques like recurrent neural networks [32].

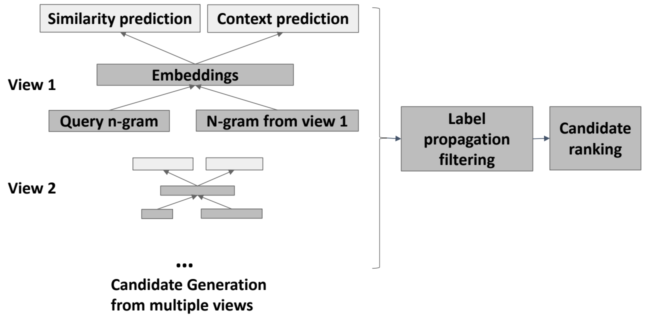

Figure 1: The proposed general framework of synonym expansion in email search, where terms are learned from different views of the data.

3.1 Learning from different views

Data sparsity is an intrinsic problem of email search [3]. To combat data sparsity, for each term, we learn multiple representations from different views, and then synthesize these representations. Specifically, we use three views: click, query session, and user distribution. The learning method is similar when modeling each of these views.

3.1.1 *Generalization of emails and queries.*

Before diving into the details of our methods, we need to point out that both emails and email search queries contain sensitive private data: email bodies are not allowed to use, while queries and email subjects can only be used with special treatment. Furthermore, email queries and emails do not generalize across users [3]. To solve these problems, we employ an attribute parameterization method proposed by Bendersky et al. [3], which enables effective usage of cross-user interactions in personal search. Specifically, we first extract all word n-grams (for n = 1, 2, 3) from queries and email subjects. Among these n-grams, we only keep frequent ones while dropping infrequent ones. We say an n-gram is frequent if it appears in query logs of sufficiently many users. Finally, we represent a query by its longest frequent n-gram and an email by a very small set of frequent n-grams appearing in the email subject. For example, the query *"bob weekly schedule"* will be represented by *"weekly schedule"*, and the email with subject *"Friday lunch invitation for Alice!"* will be represented by [*"friday lunch", "lunch invitation"*].

Formally, the list of frequent query n-grams from the entire data set is denoted by $[\mathbf{q}_1, ..., \mathbf{q}_N]$, where N is the number of frequent query n-grams. The i-th query n-gram $\mathbf{q}_i$ is composed of a sequence of words $[q_i^1, ..., q_i^{|\mathbf{q}_i|}]$, where $|\mathbf{q}_i|$ is the length of $\mathbf{q}_i$. Similarly, the frequent document n-grams from the entire data set is represented by $[\mathbf{d}_1, ..., \mathbf{d}_M]$, where M is the number of frequent document n-grams. The j-th document n-gram $\mathbf{d}_j$ is a sequence of words $[d_j^1, ..., d_j^{|\mathbf{d}_j|}]$.

3.1.2 *Lexical similarity of synonyms in email search.*

Unlike the synonyms that one can find in a dictionary, email search synonyms may often be lexical variations on the original query n-grams. For example, emails from the *H&M* clothing company are often most retrievable by term *hm* (as the company domain name is *hm.com*). Thus, for search purposes, a good synonym for *h&m* is *hm*.

Also, quite often we see users misspell their queries. E.g., we would want to add "Amazon" as a synonym to query "Amazin", in case this misspelling is not detected by a standard spelling correction system.

To account for this lexical similarity, we experimented with adding edit distance as a feature. However, this does not work well in practice because machine learning models tend to rely heavily on edit distance and ignore other signals. We found that a preferable way is to learn subword information by embedding character n-grams [7]. In this way, a word is represented by the sum of its word embedding and character n-gram embeddings. For example if we consider character n-grams of size 2 and 3, then for the word *h&m*, we not only embed *h&m* itself, but also take into account character n-grams [*␣h*, *h&*, *&m*, *m␣*, *␣h&*, *h&m*, *&m␣*], where ␣ denotes word boundaries. In experiments, we use character n-grams of size from 3 to 6.

3.1.3 *Learning from queries and email clicks.*

We first describe our model in the view of queries and email clicks. How to generalize this model into other views will be specified later on. From a high level, the model is an instantiation of the graph depicted in Figure 1. One n-gram from a query and one n-gram from an email are fed into the model, in which they go through an embedding layer, trained for two tasks: similarity prediction and context prediction. Below we give a detailed description of the two tasks.

Similarity prediction. In the context of queries and email clicks, similarity means whether a query n-gram is similar to a document n-gram. For a query and document n-gram pair, we set the ground-truth label as positive if the document n-gram is from a clicked document of this query, otherwise we set the ground-truth label as negative. We denote the embedding vector of word i by $\mathbf{z}_i$ (which already is a sum of word embeddings and character n-gram embeddings as described above). The predicted similarity $\hat{y}_k^s$ for the k-th example is:

$$\hat{y}_k^s = \phi(\mathbf{z}_{q_k^1}, \ldots, \mathbf{z}_{q_k^{|\mathbf{q}_k|}}, \mathbf{z}_{d_k^1}, \ldots, \mathbf{z}_{d_k^{|\mathbf{d}_k|}}), \tag{1}$$

where $\hat{y}_k^s \in [0,1]$, $\phi(\cdot)$ is a neural network that takes as input the embeddings of words from both the query n-gram $\mathbf{q}_k$ and the document n-gram $\mathbf{d}_k$ of the k-th example, $\mathbf{z}_{q_k^i}$ is the embedding of the i-th word q_k^i in $\mathbf{q}_k$, and $\mathbf{z}_{d_k^j}$ is the embedding of the j-th word d_k^j in $\mathbf{d}_k$.

Given a set of training data $\{\mathbf{q}_k, \mathbf{d}_k, y_k^s\}$, where $y_k^s \in \{0,1\}$ denotes no-click or click, we minimize the cross entropy loss:

$$\mathcal{L}^s = -\sum_k \Big(y_k^s \log \hat{y}_k^s + (1 - y_k^s) \log(1 - \hat{y}_k^s) \Big), \tag{2}$$

We use a multi-layer perceptron (MLP) as an instantiation of the neural network, which is fast and effective for short phrases. In our experiments, embeddings for both a query n-gram and a document n-gram are simple sum-ups of embeddings of words in the n-gram, respectively.

Note that more advanced models like recurrent neural networks [32] are not applicable to our case because of privacy concerns – we are only able to use a very limited number of frequent n-grams from email subjects, with no sequence information preserved.

Context prediction. Although *similarity prediction* learns representations of n-grams effectively, it may not learn representations of individual words equally effectively. This is because with similarity prediction, the training signal of one example (document n-gram, query n-gram) is always backpropagated to multiple words (unless the n-gram is a unigram). To bring more information into each word, we employ word2vec-style training [47]. That is, we embed each query word into the context of clicked documents and vice versa. Formally, the context $\mathbf{c}_k = \{w | w \in \mathbf{q}_k \vee w \in \mathbf{d}_k\}$ of the k-th example is a set of words from either the query n-gram $\mathbf{q}_k$ or the document n-gram $\mathbf{d}_k$ where the ground-truth similarity label y_k^s is positive. We treat any pairs of words c_k^i and c_k^j from the context as positive examples and sample negative examples randomly from the dictionary. Using the binary logistic loss, the negative log likelihood is:

$$\log\Big(1 + \exp(-s(\mathbf{z}_{c_k^i}, \mathbf{z}_{c_k^j}))\Big) + \sum_{n \in \mathcal{N}_{\mathbf{c}_k}} \log\Big(1 + \exp(s(\mathbf{z}_{c_k^i}, \mathbf{z}_n))\Big), \tag{3}$$

where $\mathcal{N}_{\mathbf{c}_k}$ is a set of negative examples sampled from the vocabulary, and the score function s is simply the inner product $s(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^{\top} \mathbf{z}_j$. Denote the logistic loss function by $\ell : x \mapsto \log(1 + \exp(-x))$, we minimize the loss for context prediction:

$$\mathcal{L}^c = -\sum_{k, y_k^s = 1} \sum_{c_k^j \in \mathbf{c}_k} \sum_{c_k^i \in \mathbf{c}_k} \Big(\ell(s(\mathbf{z}_{c_k^i}, \mathbf{z}_{c_k^j})) \sum_{n \in \mathcal{N}_{\mathbf{c}_k}} \ell(-s(\mathbf{z}_{c_k^i}, \mathbf{z}_n)) \Big), \tag{4}$$

where k is the k-th training example.

During training, the tasks of similarity prediction and context prediction are optimized jointly:

$$\mathcal{L} = \mathcal{L}^s + \mathcal{L}^c, \tag{5}$$

Summary of the model. A more detailed model structure is shown in Figure 2. A word's embedding is a sum-up of the embeddings of the word itself and its character n-grams; an n-gram embedding is a sum-up of the embeddings of its words; a query n-gram and a document n-gram are concatenated and fed through dense layers to make a similarity prediction. At the same time, similarly to word2vec, each word takes turns to predict all the other words as a context prediction task. The embedding matrix for the two tasks is shared, so that the embeddings can be simultaneously optimized by both tasks.

3.1.4 *Modeling with multiple views.*

The private nature of email search imposes several important data sparsity challenges [3]. As mentioned in Section 1, we only use very limited number of k-anonymized n-grams from email search queries and email subjects. We do not use any information from the email body, nor any sequence information from email subjects. Under these strict constraints, the information we can obtain from the *queries and email clicks* view is very sparse.

To reduce information sparsity, we extend our embedding model using a multi-view paradigm [28], which could incorporate additional data sources into our model. Specifically, we incorporate

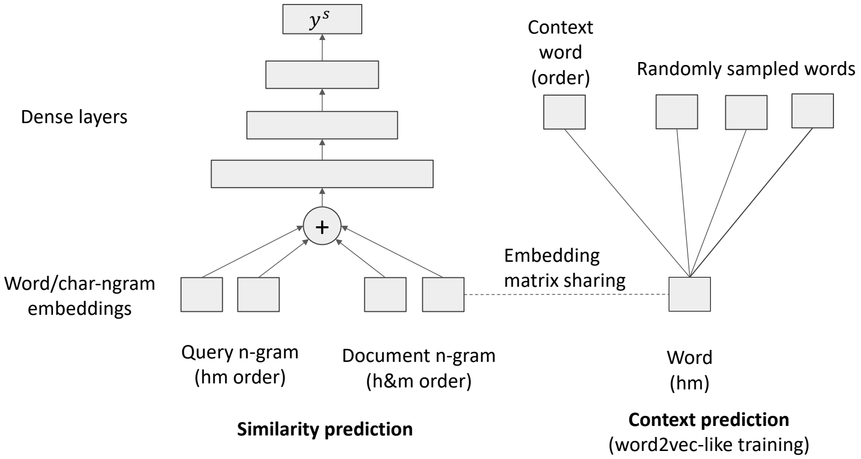

Figure 2: Model structure using the click view as an example.

two more data sources into our model, and we call them the *query session* view and the *user distribution* view.

Query session view. To build a query session view, we divide a user's search query sequence into sessions. Two queries belong to one session if the time interval between them is smaller than a threshold (which is set to 5 minutes in our experiments). Though there are more advanced methods to divide query sequences into sessions [38], they are difficult to apply in practice in email search, due to the aforementioned privacy constraints. We also find that our simple session definition works well for the email search scenario.

Our query session view is built on the assumption that queries in the same search session are very likely to carry the same search intent, which could be useful for finding synonyms. To use query session information in our model, for the similarity prediction task, when two query n-grams occur in the same search session, the label is 1; otherwise, the label is 0. For the context prediction task, each query word is embedded into the context of words from query n-grams that are from the same session.

User distribution view. The intuition of the user distribution view comes from recommender system research [51] – similar users issue similar queries. Following this intuition, we project both users and queries into an embedding space, and measure their similarity in this space. Specifically for the similarity prediction task, when an n-gram is issued as part of a search query by a user, the label is 1; otherwise, the label is 0. For the context prediction task, we embed each query word into the context of users who issued this query.

Using user distribution for similarity prediction can be connected to the Matrix Factorization model [40], which seeks to find a low rank approximation to the sparse query-user matrix by minimizing the cross entropy loss on the known elements. The query-user matrix sets a known element (q, u) to value 1 if a user u issued a query that contains n-gram q.

We summarize the similarity and context prediction tasks of all the three views in Table 1.

3.1.5 *Candidate generation.*

We return a merged list of candidate synonyms from all of the three views. Specifically, for a query n-gram, the candidate synonyms are the top K n-grams that lie closest in an embedding space of each of the views, as its candidate synonyms. The closeness metric is simply defined by the cosine similarity of the n-grams in the embedding space.

3.2 Label propagation filtering

Synonyms found by the multi-view embedding models may be noisy if directly used for query expansion. One problem is that semantically similar n-grams may not necessarily be good synonyms. E.g., n-grams like *Amazon shipping* and *Ebay shipping* are semantically similar, but adding *Ebay shipping* as a synonym to *Amazon shipping* would most often lead to bad user experiences.

As the problem is caused by misalignment between semantic similarity and query expansion for search, it can be solved by taking into account search behavior. Specifically, we can count neighbor similarity between n-grams in a click weighted graph. Figure 3 shows such a graph built between query n-grams and clicked document n-grams. The intuition is that synonyms are closer and share more neighbors in such a graph.

One option is to add a neighbor-similarity based loss during the process of embedding learning, like the method proposed in [10]. But as a soft constraint, neighbor-similarity based loss still leads to noise due to the tradeoff between itself, the *similarity prediction* loss and *context prediction* loss presented earlier. Therefore, we choose to apply a hard, post embedding training filter based on neighbor-similarity. For the example shown in Figure 3, *Amazon shipping* shares more neighbors with *Amazon tracking* than with *Ebay shipping*, and is therefore closer to *Amazon tracking*.

Specifically, we use label propagation to measure neighbor similarity between two n-grams in the bipartite graph [4, 60]. Label propagation is a graph-based semi-supervised learning algorithm that iteratively propagates labels from nodes to neighboring nodes. To do so, we create an edge and assign edge weight from n-gram a to n-gram b based on the number of clicks, and we set the initial label of each node as the n-gram it represents. After label propagation, each node in the graph will be labeled by a set of propagated labels

Table 1: The summary of different views.

View	Similarity	Context
Click	Subject n-gram is clicked for a query n-gram	Embed a query n-gram with the clicked subject n-grams
Query session	Two query n-grams are in the same search session	Embed a query n-gram with query n-grams in the same session
User distribution	User issued a query n-gram	Embed a query n-gram with the users who issued this query n-gram

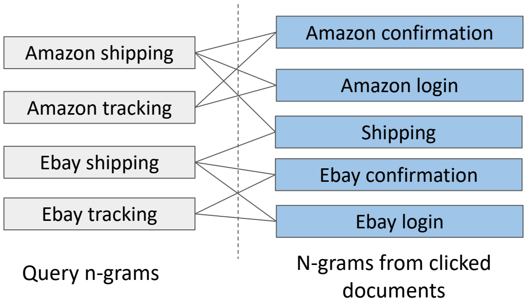

Figure 3: Noise filtering based on the bipartite graph of query-clicks.

(in our case, n-grams), together with label weights. This enables us to measure similarity of two n-grams by cosine similarity of their labels. Now given a query n-gram and a candidate synonym, we discard the synonym if similarity is nonpositive.

3.3 Candidate ranking

In order to make the results more robust, we ensemble the results learned from each view by learning-to-rank techniques. To create ground-truth labels for rankers, we randomly sample a set of query n-grams, and manually label the relevance of their synonym candidates pooled from all three learning views. The features fed to a ranker are cosine similarities of embedding vectors of query n-grams and candidate synonyms from each view, cosine similarity values based on propagated labels, and two additional properties for each synonym candidate. The two properties are: (1) the issue frequency, and (2) the number of emails returned in the past month to queries containing the synonym candidate. The objective is to promote the ranking of synonym candidates with higher relevance.

4 EXPERIMENT SETUP

This section describes our setup for the competing methods, the data sets, and the evaluation metrics. We first describe the setting for offline experiments, followed by online ones.

4.1 Baselines

For offline experiments, we compare with a list of baselines.

WordNet. In order to compare with synonyms derived from external resources, we utilize WordNet [29, 45, 58], where "synsets" can be used as synonyms. Following [1], for n-grams with $n > 1$, multiple synonyms are created by substituting one of the words in the n-gram with one of its synonyms from WordNet.

There might be other external resources where synonyms can be mined. However, comparing synonyms from general external resources with synonyms derived from email search is not the focus of this paper. The reasons are twofold. First, there is a shift in semantics from email search to general scenario search, resulting limited applicability of general external resources. Recall the *target* example we have in Section 1, where *target* in email search most often refers to the store, rather than *goal*. Second, in our online experiment, we measured improvement over Google's web synonyms, which is already a very strong baseline of general synonyms.

DESM. The Dual Embedding Space Model (DESM) [49] trains an embedding model on the Bing query corpus. The trained embeddings are publicly available[1]. We report the performance of the IN-OUT configuration of the DESM model, which has the best performance in the original paper.

Label propagation. Query expansion using query-click bipartite graphs has been proven to be an effective method [2, 16]. As users do not share their private documents, instead of building the graph on clicked document, we build the graph between frequent query n-grams and frequent document n-grams, where label propagation can be performed. This follows the same procedure as described in Section 3.2.

Clicks, **Users**, **Sessions**. Individual embedding models learned from each view represent strong baselines that utilize the information of clicks (or relevance feedback), user interactions, and search sessions, respectively.

Rank fusion. Instead of using learning-to-rank techniques to join information from different views, we employ an unsupervised method – Reciprocal Rank Fusion (RRF) [17], which has been shown to be very effective in combining the rankings from multiple IR systems.

4.2 Variants of our methods

To use learning-to-rank methods to ensemble the results from different views, we experiment with all methods provided by the RankLib library. [2] These methods serve as different instantiations of our synonym expansion framework. We use the names of the learning-to-rank methods to refer to different instantiations. They are **LambdaMART** [61], **RankBoost** [26], **MART** [27], **ListNet** [12], **Random Forests** [9], **AdaRank** [63], **Coordinate Ascent** [46], and **RankNet** [11].

As noted before, we use MLP as our neural network model. Due to privacy concerns, more advanced models like RNNs [32] are not

[1] https://www.microsoft.com/en-us/research/project/dual-embedding-space-model-desm

[2] https://sourceforge.net/p/lemur/wiki/RankLib/

applicable to our case – we can only obtain a few frequent n-grams from email subjects, without preserving sequence information.

4.3 Data sets

We use one month of search logs obtained from Gmail consumers. The synonyms found by all methods (except for WordNet and DESM) are derived from this log. The raw queries contain 1.56 words on average. From each raw query, we extract at most one frequent n-gram (if there is), whose average number of words is 1.25. The number of n-grams extracted from each email subject is 2.31 on average. As noted before, email body cannot be used.

To judge the relevance of candidate synonyms, some work uses these synonyms to rerank documents. This is not an ideal option for us, as we want to evaluate whether the newly found synonyms can be used to retrieve more relevant documents, rather than reranking retrieved documents. However due to privacy issues, we cannot annotate the relevance of newly retrieved documents manually. So we decided to evaluate the effectiveness of synonyms in the retrieval system only through online experiments. In the offline setting, we use human experts to judge the relevance of query-synonym pairs.

The annotation process is as follows. We randomly sample 250 frequent query n-grams, and pool synonyms of these n-grams proposed by all the methods. We first set aside 10 queries (which contain about 200 query-synonym pairs) for four human experts to assess rater agreement. We compute Cohen's kappa between all pairs of raters, and reach a Kappa score of 0.62105 on average. Scores that fall between 0.61 and 0.80 are described as substantial agreement. Since this agreement level is acceptable, we assign each rater a set of queries so that all queries will be judged by three raters. We exclude non-English queries and queries without positively labeled synonyms, resulting in a final set of 5,859 synonyms (which belong to 177 unique query n-grams). The label of a synonym is the sum of binary labels given by each rater. So the label value ranges from 0 to 3, allowing us to use metrics that consider graded relevance.

4.4 Evaluation metrics

We use Normalized Discounted Cumulative Gain (**NDCG@5**) as our main evaluation metric for offline evaluation [36]. This popular IR metric considers both the rank of relevant documents in the predicted order and their graded relevance. To gain further insight into the behavior of methods, we calculate **precision** and **recall**.

4.5 Hyper-parameter setting

Since we are working on an unsupervised task, we do not have an evaluation set to tune hyper-parameters. Therefore, all the hyper-parameters are set to their recommended values. The MLP neural network used in our model uses tanh as the activation function, and has three dense layers of size 64, 32 and 16, respectively. It optimizes towards a loss function specified in Eq. 5 using AdaGrad [23] as the optimizer. The embedding size is set to 64. The initial learning rate is set to 0.05. For character n-grams, we use n from 3 to 6. For each query, we select up to top $K = 10$ candidate synonyms from each of the competing methods.

4.6 Setup for online experiments

Online experiments allow us to evaluate the true effectiveness of the found synonyms by observing user click behavior on retrieved documents. When these (*query n–gram*, *synonym*) pairs are served in live experiments, a synonym will be automatically added to a query if the query contains the query n-gram. We select two methods for live experiments – the strongest baseline and our proposed method in its strongest configuration.

All the live experiments include the synonyms used by Google Web Search [43]. This is a comprehensive list of synonyms developed for decades at Google; the synonyms were derived by various proprietary state-of-the-art algorithms that are based on clicks, user sessions, etc. This ensures that even if no new synonyms are added by our methods, synonyms found by web search, if present, will still be added to a query. This setting (a) guarantees that our model is competitive in a production setting, and (b) validates our hypothesis that our synonyms are sufficiently different than those that could be mined from the web corpus.

We will then have three groups of experiments: (1) the control group, where only web-based synonyms are added; (2) treatment group 1, which includes synonyms from the web and $4,000$ top synonyms ordered by the score of the strongest baseline; (3) treatment group 2 with synonyms from the web and top $4,000$ synonyms from our proposed method. Each group will be tested separately on a large amount of traffic of the Gmail Search service.

The metrics used in live experiments are Click-through Rate (CTR) and Mean Reciprocal Rank (MRR), which are commonly used metrics for email search [3]. Given N queries, MRR is defined as:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \tag{6}$$

where rank_i is the rank of the first clicked document of the i-th query.

Note that we are limited to a single-click metric (MRR), rather than a multi-click metric like NDCG. This is due to the nature of our search setting – Gmail search overlay presents a ranked list of search results as the user types, and a single click on any of the results will take the user from the search overlay to the clicked email, concluding the search session. Hence MRR is used as the key metric in various studies for Gmail Search [3, 59, 65].

5 EXPERIMENT RESULTS

5.1 Overall performance in offline experiment

We randomly divide our annotated dataset into 10 folds. Each fold is used once as the test set, with the remaining 9 folds compromising the training set, and we report the performance averaged over 10 folds. The overall performance is shown in Table 2.

We first focus on NDCG@5, the major metric that compares performance. All configurations of our proposed methods outperform the baselines. This shows that utilizing labeled data to learn to join information from different data sources could bring positive effects. *Rank fusion* works better than individual models, meaning that combining information from different views is a successful way to combat data sparsity in email search. Comparing between methods *Clicks*, *Users* and *Sessions*, which are embedding models trained on a

Table 2: Overall performance in offline experiment. "*" indicates that NDCG@5 of a method is statistically significantly better than Rank fusion, according to t-test at levels of 0.01.

	NDCG@5	Precision	Recall
WordNet	0.3141	0.1880	0.0598
DESM	0.3413	0.2018	0.0837
Label propagation	0.3723	0.3322	0.2831
Clicks	0.5947	0.5338	0.4601
Users	0.5848	0.5586	0.3659
Sessions	0.5089	0.4841	0.1973
Rank fusion	0.6136	0.5224	**0.6744**
MART	0.6823*	0.5224	**0.6744**
RankNet	0.6744*	0.5224	**0.6744**
RankBoost	**0.7031***	0.7238	0.4616
AdaRank	0.6826*	**0.7779**	0.3862
Coordinate Ascent	0.7011*	0.7400	0.6227
LambdaMART	0.6892*	0.7664	0.5042
ListNet	0.6722*	0.5224	**0.6744**
Random Forests	0.6714*	0.5224	**0.6744**

single source, *Sessions* perform much worse than the other two. This is mainly due to the sparsity of session information in email search. Embedding models outperform *Label propagation* by a large margin, indicating the effectiveness of our design choice of embedding models, which optimize for both similarity and context prediction, and consider syntactic and semantic similarity simultaneously.

Looking at methods using external information, *WordNet* performs the worst, as it cannot handle new words, misspelled words, and words that have changed meaning in email search. For example, "virago" might be a good synonym for "amazon" in a more general context. However in email search, recommending "virago gift" as a synonym for "amazon gift" does not make much sense. DESM, which is an embedding model trained on Bing query corpus, performs better than WordNet, but is worse than models trained specifically on the Email data set. It suggests good candidates when a query is also frequent in web search. For example, it proposes "gps" as a synonym for query "garmin". However, when it comes to more email related queries, such as "uber receipt", it often fails to suggest good synonyms.

Now we turn our attention to precision and recall. Like *Rank fusion*, *MART* gives positive prediction to all examples, thus obtaining the same metric scores. As *Rank fusion* and learning-to-rank methods mainly optimize ranking instead of binary relevance, it will be more interesting to look at other methods. The reasonable precision but low recall of *Sessions* confirms our prior statement on session sparsity. *WordNet* suffers severely from the recall problem, as it cannot provide synonyms for new words or misspelled words that appear in the private corpora.

5.2 Overall performance in online experiment

Many learning-to-rank techniques achieve comparably good performance in offline experiments. We use *Coordinate Ascent* as our representative method for online experiments, as its variance of performance on different folds is smaller than other methods.

Table 3: Overall performance in online experiment. "*" indicates that a method is statistically significantly better over web synonyms, according to t-test at levels of 0.05. "+" means that we are measuring the relative improvement over web synonyms.

	MRR	CTR
Rank fusion	+0.97	+1.56*
Coordinate Ascent	+1.32*	+1.68*

Baseline. In addition to *Rank fusion*, which is the strongest baseline in offline experiments, we also consider the synonyms developed for Google web search. It is worth noting that Google web synonyms form a very strong baseline, as it includes a huge number of synonyms developed over decades in Google using various proprietary state-of-the-art algorithms.

Table 3 gives the summary of the online experiment, which shows the relative improvement over Google web synonyms. Both *Rank fusion* and *Coordinate Ascent* are able to outperform web-based synonyms. This indicates that synonyms found by our proposed method are not only reasonable by the offline judgment of human raters, but are also effective in surfacing more relevant documents in real settings. It also suggests that web based synonyms are not sufficient in email search, where models specifically designed for this scenario are necessary. Comparing between *Coordinate Ascent* and *Rank fusion*, the former has higher gains, especially in terms of MRR. This again confirms that using learning-to-rank techniques to combine information from different views is beneficial.

5.3 Feature analysis by ablation study

In order to analyze the importance of individual feature categories, we perform an offline ablation study. Again we use *Coordinate Ascent* for the following analysis due to its smallest variance of performance on different folds.

Features are categorized into five groups: (1) model score of *Clicks*; (2) model score of *Users*; (3) model score of *Sessions*; (4) similarity score of *Label propagation*, and (5) features related to synonym properties. Synonym properties are described in detail in Section 3.3.

In the ablation study, we use a single feature group alone once at a time, with all other features excluded. The results are displayed in the first column of Table 4, which show the percentage of relative performance reduction comparing to the method using all feature groups. Similarly, we also remove one feature group at one time, while keeping all other features present. This is shown in the second column of Table 4.

All numbers in Table 4 being positive indicates that all feature groups contribute positively to the task. By integrating information from different sources, our methods are able to alleviate the sparsity problem in email search. The reduction in the second column of Table 4 is relatively small, i.e., some information might be shared across sources. Due to the small values in the second column, the following analysis will be based on the first column of Table 4.

The performance of *Clicks*, *Users* and *Sessions* are in general consistent with their performance in Table 2.

Table 4: NDCG@5 reduced by including one feature group alone or by excluding a feature group.

Feature group	Include one group	Exclude one group
Clicks	9.44%	1.17%
Users	13.64%	0.78%
Sessions	26.61%	1.92%
Label propagation	16.12%	0.84%
Properties	24.18%	0.75%

Comparing to its poor performance in Table 2, *Label propagation* gives a smaller reduction of NDCG. This is because the similarity score of *Label propagation* itself might promote some irrelevant terms as synonyms. However, it also attaches (usually lower) scores to examples found by embedding-based models. With labels, learning-to-rank methods learn to find a function that promotes truly relevant terms to the top.

5.4 Synonym examples

We cannot disclose a comprehensive list of synonyms produced by our model, due to the proprietary nature of the data. Therefore, in this section we provide a brief summary of the types of the generated synonyms, with a few illustrative examples.

Recall that our model uses both word and character level embeddings (see Figure 2). Therefore, it is able to produce both *syntactic* and *semantic* synonyms.

As examples of syntactic synonyms, we observe multiple types of common syntactic modifications: word merge and splits, completion, and word stemming. As an example, *blablacar* ⟶ *bla bla car* is a case of word split. We also observe cases of character substitution, e.g., *whentowork* ⟶ *when2work*. Such synonyms can be very useful when the user cannot recall an exact spelling of a service provider, a very common scenario in email search.

For semantic synonyms, we find candidates that, while not close in their surface forms, are highly related to the original query, and may help in increasing retrieval recall when the user does not remember the exact wording of the desired email message, e.g., *uber receipt* ⟶ *uber invoice* or *company recruiting* ⟶ *company interview*.

6 CONCLUSION

In this paper, we consider the task of synonym expansion in email search. Synonym expansion adds related words to search queries, and can improve recall by retrieving more relevant documents. Though it has been extensively studied for web search, few studies have tackled this problem for email search, where unique research challenges exist. The documents are not shared across users, user search queries may not be generalizable across users, and the sizes of the private corpora are much smaller than that of the web corpus.

We propose a solution to address the challenges of synonym expansion for private corpora. We view the search log from multiple perspectives by representing a query by its clicked documents, a set of users who issued them, and a set of queries that are in the same search session. A novel embedding-based framework is then developed to learn and join information from the multiple sources to obtain optimal synonym candidates. To demonstrate the effectiveness of the proposed technique, we evaluate our model using both explicit human ratings as well as a live experiment on Gmail search, one of the world's largest email search engines. The live experiment demonstrates significant improvements over Google's state-of-the-art production baseline.

REFERENCES

[1] Rao Muhammad Adeel Nawab, Mark Stevenson, and Paul Clough. 2012. Detecting text reuse with modified and weighted n-grams. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. 54–58.

[2] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics* 40, 1 (2014), 57–84.

[3] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from user interactions in personal search via attribute parameterization. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 791–799.

[4] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label Propagation and Quadratic Criterion. *Semi-Supervised Learning* (2006).

[5] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query Suggestions in the Absence of Query Logs. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 795–804.

[6] Abhijit Bhole and Raghavendra Udupa. 2015. On Correcting Misspelled Queries in Email Search. In *Association for the Advancement of Artificial Intelligence*. 4266–4267.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* (2016).

[8] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. 531–541.

[9] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[10] Thang D Bui, Sujith Ravi, and Vivek Ramavajjala. 2017. Neural graph machines: Learning neural networks using graphs. *arXiv preprint arXiv:1703.04818* (2017).

[11] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*. 89–96.

[12] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*. 129–136.

[13] 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 Conference on Information and Knowledge Management*. 283–292.

[14] David Carmel, Liane Lewin-Eytan, Alex Libov, Yoelle Maarek, and Ariel Raviv. 2017. The demographics of mail search and their application to query suggestion. In *Proceedings of the 26th International Conference on World Wide Web*. 1541–1549.

[15] Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)* (2012), 1:1–1:50.

[16] Kevyn Collins-Thompson and Jamie Callan. 2005. Query expansion using random walk models. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. 704–711.

[17] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval*. 758–759.

[18] W Bruce Croft and David J Harper. 1979. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation* (1979), 285–295.

[19] W. B. Croft and D. J. Harper. 1988. Document Retrieval Systems. Taylor Graham Publishing, Chapter Using Probabilistic Models of Document Retrieval Without Relevance Information, 161–171.

[20] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2003. Query expansion by mining user logs. *IEEE Transactions on Knowledge and Data Engineering* 15, 4 (2003), 829–839.

[21] Dotan Di Castro, Liane Lewin-Eytan, Yoelle Maarek, Ran Wolff, and Eyal Zohar. 2016. Enforcing k-anonymity in web mail auditing. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. 327–336.

[22] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 1929–1932.

[23] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[24] Efthimis N Efthimiadis. 1996. Query Expansion. *Annual Review of Information Science and Technology (ARIST)* (1996), 121–87.

[25] Bruno M Fonseca, Paulo Golgher, Bruno Pôssas, Berthier Ribeiro-Neto, and Nivio Ziviani. 2005. Concept-based interactive query expansion. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management.* 696–703.

[26] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* 4, Nov (2003), 933–969.

[27] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

[28] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. 2014. A multi-view embedding space for modeling internet images, tags, and their semantics. *International Journal of Computer Vision* 106 (2014), 210–233.

[29] Zhiguo Gong, Chan Wa Cheang, et al. 2006. Multi-term web query expansion using WordNet. In *International Conference on Database and Expert Systems Applications.* 379–388.

[30] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 383–392.

[31] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to Rewrite Queries. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management.* 1443–1452.

[32] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[33] Michal Horovitz, Liane Lewin-Eytan, Alex Libov, Yoelle Maarek, and Ariel Raviv. 2017. Mailbox-Based vs. Log-Based Query Completion for Mail Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 937–940.

[34] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the Association for Information Science and Technology* 54 (2003), 638–649.

[35] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management.* 2333–2338.

[36] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[37] Karen Sparck Jones. 1971. Automatic keyword classification for information retrieval. *Archon Books* (1971).

[38] Rosie Jones and Kristina Lisa Klinkner. 2008. Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management.* 699–708.

[39] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web.* 387–396.

[40] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009), 30–37.

[41] 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.

[42] Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* 1929–1932.

[43] John Lamping and Steven Baker. 2009. Determining query term synonyms within query context. US Patent 7,636,714.

[44] Xiujun Li, Chenlei Guo, Wei Chu, Ye-Yi Wang, and Jude Shavlik. 2014. Deep learning powered in-session contextual ranking using clickthrough data. In *In Proc. of Conference on Neural Information Processing Systems.*

[45] Rila Mandala, Tokunaga Takenobu, and Tanaka Hozumi. 1998. The use of WordNet in information retrieval. *Usage of WordNet in Natural Language Processing Systems* (1998).

[46] Donald Metzler and W Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval* 10, 3 (2007), 257–274.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[48] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web.* 1291–1299.

[49] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137* (2016).

[50] Yonggang Qiu and Hans-Peter Frei. 1993. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 160–169.

[51] Paul Resnick and Hal R Varian. 1997. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–58.

[52] Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics.* 464–471.

[53] Joseph John Rocchio. 1971. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing* (1971).

[54] Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *SIGIR Workshop on Neural Information Retrieval* (2016).

[55] Gerard Salton and Chris Buckley. 1997. Readings in Information Retrieval. Chapter Improving Retrieval Performance by Relevance Feedback, 355–364.

[56] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management.* 101–110.

[57] Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. 2014. Learning Concept Embeddings for Query Expansion by Quantum Entropy Minimization. In *Association for the Advancement of Artificial Intelligence.*

[58] Ellen M Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 61–69.

[59] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining.* 610–618.

[60] Yan Wang, Rongrong Ji, and Shih-Fu Chang. 2013. Label propagation from imagenet to 3d point clouds. In *2013 IEEE Conference on Computer Vision and Pattern Recognition.* 3135–3142.

[61] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13, 3 (2010), 254–270.

[62] Jinxi Xu and W Bruce Croft. 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems* (2000), 79–112.

[63] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 391–398.

[64] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 323–332.

[65] 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.

[66] Hamed Zamani and W Bruce Croft. 2016. Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval.* 147–156.

[67] Hamed Zamani and W Bruce Croft. 2017. Relevance-based Word Embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 505–514.

[68] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 575–584.