Learning Effective Embeddings for Machine Generated Emails with Applications to Email Category Prediction

Yu Sun[*] Twitter ysun@twitter.com
Lluis Garcia-Pueyo[*] Facebook lgp@fb.com
James B. Wendt Google jwendt@google.com
Marc Najork Google najork@google.com
Andrei Broder Google broder@google.com

***Abstract*—Machine generated business-to-consumer (B2C) emails such as receipts, newsletters, and promotions constitute a large portion of users' inboxes today. These emails reflect the users' interests and often are sequentially correlated, e.g., users interested in relocating may receive a sequence of messages on housing, moving, job availability, etc. We aim to infer (and eventually serve) the users' future interests by predicting the categories of their future emails. There are many useful methods, such as recurrent neural networks, that can be applied for such predictions, but in all cases the key to better performance is an effective representation of emails and users. To this end, we propose a general framework for learning embeddings for emails and users, using as input only the sequence of B2C *templates* users receive and open. (A *template* is a B2C email stripped of all transient information related to specific users.) These learned embeddings allow us to identify both sequentially correlated emails and users with similar sequential interests. We can also use the learned embeddings either as input features or embedding initializers for email category prediction tasks. Extensive experiments with millions of fully anonymized B2C emails demonstrate that the learned embeddings can significantly improve the prediction accuracy for future email categories. We hope that this effective yet simple embedding learning framework will inspire new machine intelligence applications that will improve the users' email experience.**

I. Introduction

One common question raised in many applications including recommendation systems, digital assistants, and personalized content delivery is what users will be interested in next. We aim to answer this question with users' email data. Every day, millions of business-to-consumer (B2C) emails, including coupons, booking confirmations, payment receipts, etc., are sent to users. These B2C emails often reflect user interests, e.g., users interested in investing may receive many emails on financial market analysis, while others who enjoy online shopping may receive many coupon emails and shipping notifications. Causal threads [1] also exist among the emails users receive, e.g., for a travel thread users may first receive emails regarding flight tickets, then a hotel booking confirmation, followed by car rental receipts. We propose to exploit the B2C emails users receive and the correlation among these emails to predict which B2C emails users will receive next, and hence discover what users will be interested in next.

[*] Work done while at Google.

A recent study conducted by Yahoo! Labs reports that 90% of today's non-spam Web Mail traffic is machine generated [1]. The majority of B2C emails are also created by automatic scripts, i.e., by populating a fixed *template* with transient components showing, e.g., the name of customers, shipping date, amount of payment, etc. [2], [3]. By clustering similar B2C emails together and removing the transient components, we can extract the underlying template. We propose to use the extracted template to replace the actual emails, because this not only protects user privacy, but also retains a higher signal-to-noise ratio since the templates summarize many emails and are less noisy.

Given that the template category (e.g., Travel or Sports News) is an even higher-level abstraction of various templates, in this paper, we focus on predicting the *categories* of future email templates users will receive. In particular, the first prediction task we study is (i) to predict users' email categories in a future time window, say three days, while the second (ii) is to predict the next email category that users will open and read.

The email category prediction tasks are of particular interests because they have a wide range of potential applications, including improved user experiences (e.g., warning users of items ordered but not shipped), targeted advertising (users who recently made a flight reservation may be interested in hotel reservations), and spam classification (an email that is part of a legitimate causal thread is unlikely to be spam). We further investigate the second task because recent studies reveal that many B2C emails are unread or deleted without being read [4], [5] and reading suggests a stronger interest.

To solve the above two prediction tasks, many predictors and classifiers such as neural networks can be used, where the prediction accuracy largely relies on the input feature representations. Therefore, in this paper, we focus on (i) obtaining effective representations for email templates, template categories and users through a simple, general and unified embedding learning framework, and (ii) investigating the performance enhancement such representations can bring to the above two email category prediction tasks. We are interested in a general representation of the templates and users such that they can also be used in many other downstream tasks including identifying and bundling together emails of the same causal thread and finding users with similar interests.

The way we generate the template and user representations is inspired by the embedding learning framework that generalizes the word embedding approach [6] and focuses on items and their contexts [7]. Specifically, with users' email data, we first (i) generate template and user sequences from which we obtain contexts for each template/user, and then (ii) adapt the word embedding approach [6], [8], which uses a single-layer neural network and negative sampling, to learn the embeddings. With the template embedding, we can discover templates that are similar or sequentially correlated by finding their nearest neighbors in the embedding space. For example, in our experiments the top-3 nearest neighbors of the Travel category are Hotel, Travel Insurance, and Airport Parking (cf. Table II for more case studies). Similarly, we can also find users with similar sequential interests with the user embedding. We solve the two email category prediction tasks with long short-term memory (LSTM) and multilayer perceptron (MLP) architectures, respectively. Empirical experiments demonstrate that, when we use the learned embeddings as input, the predictive performances for the two tasks are significantly improved especially for less frequent but more important categories such as Hotel and Car Rental. The main contributions of this paper are summarized as follows.

- We propose a general embedding learning framework for generating representations of templates, template categories, and users based on email data. With the learned embeddings, we can discover sequentially correlated templates or users with similar sequential interests.
- We investigate different ways to use the learned embeddings with LSTM and MLP for email category predictions and establish that it is effective to use the learned embedding either as input features or as embedding initializers.
- We conduct extensive experiments with millions of B2C emails. The results confirm that with the learned embeddings we can find interesting clusters containing sequentially correlated templates or users having similar sequential interests. The results also show that the learned embedding can significantly improve the predictive performances for email category predictions.

II. Related Work

A. Obtaining Email Templates

Inferring the templates of machine generated B2C emails is an active area of research. Ailon et al. [1] propose finding such templates by recognizing frequent word sub-sequences in the email subjects sent by the same source, e.g., *Your order #xxx-xx is on its way*. Avigdor-Elgrabli et al. [2] extend this approach, identifying templates by clustering emails based on their HTML structure. Proskurnia et al. [9] propose inducing templates from plain text emails using suffix arrays.

There are also many email-related applications utilizing such templates. For example, Wendt et al. [10] and Potti et al. [11] exploit templates for various email classification tasks. Sheng et al. [12] use induced templates to extract structured data from email, such as Hotel Confirmations and Bill Reminders. Bendersky et al. [13] exploit the template structure and associations between templates and query terms to improve searching over personal emails.

B. Email Prediction

An important application with email templates is to predict users' future emails. Ailon et al. [1] propose to identify B2C emails that are part of a *causal thread*, e.g., *receipts for online shopping $\rightarrow$ a shipment notification $\rightarrow$ item tracking emails*. They propose to build a causal graph for the templates where edges indicate potential causal relations and then use a learned causal relation function to identify future emails' threads. Inspired by the causal threads between B2C emails, Gamzu et al. [14] further study the problem of predicting which email templates users will receive in future time windows. Their technique first finds the causality relations between templates and then with the uses these relations to build a generative model which is similar in spirit to a k-dependent Markov chain [15]. Since there could be millions of different templates while only thousands of template categories, the problem studied by Gamzu et al. is harder and different than the email category predictions we concentrate on. The closest work related to ours is the work of Zhang et al. [15], where they investigate using neural networks such as LSTM and MLP to predict the email categories users will receive in future time windows. We will introduce their work in detail in Section V-A and compare their approaches in our experiments.

Another thread of work is to predict user actions on received emails. Di Castro et al. [4] study the problem of predicting whether a user will *read*, *reply*, *forward*, or *delete* a received email using both local features related to the email and inbox and global features related to the sender (e.g., reply ratio of emails sent from `info@twitter.com`). They integrate vertical learning (i.e., for a specific user) with horizontal learning (i.e., for a cluster or all users) to address the skewed distribution of the number of training examples from heavy and light users. The difference between our work and that of Di Castro et al. [4] is that we aim to predict the arriving category of a future unseen email, for which we cannot obtain either the local or global features. In other words, the email category prediction is before the arrival of an email, while predicting the action is after and the arrived email is used as model input. The integration of vertical and horizontal learning is in principle similar to our approach of using a pre-trained embedding in the sense that they both attempt to transfer the information of some prior representations and aim to alleviate the imbalance of training data for heavy/frequent and light/rare users/categories.

C. Embedding Representation

The focus of our work is to learn effective representations of email templates and users that can be effectively used in the email category prediction tasks. Obtaining an effective representation for categorical features, words, or items has been actively studied for decades [16], [17]. Many widely used techniques such as matrix or tensor factorization [18], [19], topic modeling [20], and graph embedding [21] can be thought of as approaches for obtaining such representations. The recent success of word embedding [6], [8], [22] and image captioning [23] draws special attention to the approach of learning representations using neural networks. The key methodology is to use vectors of the (last) hidden layers as the learned representations. The word embedding generation method proposed by Mikolov et al. [6], [8] uses a word to predict its surrounding words or its *context* with a single layer feed-forward network and train the embedding over a large corpus with efficient training techniques such as hierarchical softmax or negative sampling. Rudolph et al. [7] generalize the above approach to a general embedding learning framework focusing on an item and its context. Our embedding generation framework resembles the word embedding approach and focuses more on constructing the input features (i.e., contexts of items).

III. Preliminaries

A. Email Template

A large amount of B2C emails are created by filling a fixed *template* with transient components related to individual users (e.g., user names). The underlying templates can be extracted by clustering similar emails and matching the fixed components in the same cluster (cf. Section II-A for detailed discussions). Using the templates instead of actual emails has several advantages: (i) Since a template is a generalization of a large amount of emails, the inferred category of a template, i.e., the majority category of the represented emails, is often more accurate. (ii) The signal-to-noise ratio is higher, since templates are normally only extracted when the size of a cluster is above a chosen threshold (say 100) and hence random and noisy emails are automatically filtered. (iii) Using templates protects users' privacy since all sensitive information (e.g., names, addresses) is removed. In this paper, we will use *emails* and *email templates* interchangeably since we use templates to represent all actual emails.

B. Problem Definition

We study the problem of obtaining a dense and continuous representation of B2C email templates, template categories, and users from email data and effective ways of applying such representations to email category predictions. Herein, we simply use *emails* to refer to B2C emails since all the emails we focus on are B2C emails. For email category predictions, we are interested in the categories of future emails users will receive and, one step further, will open and read. In particular, we focus on two types of prediction tasks:

(i) **Receiving-based prediction**. Given users' previously received emails, predict the email categories users will receive during the *next few days* (which many contain multiple emails).

(ii) **Interest-based prediction**. Given users' previously *opened* email categories, predict the next email category (only the next one) users will open and read.

Formally, let each email template e have three attributes: a category $c(e) \in \mathcal{C}$ where $\mathcal{C}$ is predefined and fixed, a received timestamp $t(e) \in \mathbb{R}$, and an indicator $o(e) \in \{0, 1\}$ of whether it was *opened* or not where 1 indicates opened.

(i) **Receiving-based prediction**. Given a collection of h received emails $\{e_1^u, e_2^u, \ldots, e_h^u\}$ for a user u with $t(e_i^u) \leq t_p$, predict the category $c(e^u) \in \mathcal{C}_\Delta$ in the set of email categories $\mathcal{C}_\Delta = \{c(e^u) \,|\, t(e^u) \in (t_p, t_p + \Delta]\}$ user u receives within time window Δ (e.g., three days) starting from the prediction time t_p.

(ii) **Interest-based prediction**. Given a collection of s opened emails categories $\{c(e_1^u), c(e_2^u), \ldots, c(e_s^u)\}$ with $o(e_i^u) = 1$, predict the category $c(e_{s+\delta}^u)$ of future emails $e_{s+\delta}^u$ that user u will open and read, i.e., $o(e_{s+\delta}^u) = 1$ for $\delta \in \mathbb{N}^+$.

IV. Embedding Learning

In this section, we first discuss the desired properties of the embedding and then introduce the framework for embedding learning. To the best of our knowledge, this is the first work that studies template and user embeddings derived from email data.

A. Desired Properties of the Representation

To obtain a dense and continuous representation, we project the templates/users into a continuous vector space and represent the templates/users as embedding vectors. In the embedding space, we want the template embedding vectors to have the following properties: (i) Templates of similar functionality or category, e.g., food templates about Chinese, Japanese and Vietnamese cuisine should be close to each other. (ii) Sequentially correlated templates should also be close to each other. For example, templates about flight tickets should be close to those about hotel bookings. We require this property because such sequential correlation is essential to generate high quality email category predictions.

We also want the user embedding to have similar properties: (i) users interested in common email categories (e.g., Investing, Shopping, Traveling) should be close to each other and (ii) users who have similar sequential interests, i.e., receiving and opening similar emails after reading similar previous emails, should also be in close proximity. Another property for the template/user embedding learning is (iii) that the learning process should be efficient and can easily scale to a large number of templates/users.

B. Embedding Learning Framework

Inspired by the representation learning method [7] that generalizes the word embedding approach [8], [6], [22], we develop the following embedding learning framework for templates, template categories, and users with the desired properties.

(i) **Obtaining item sequences**. From users' email data, we first obtain sequences of templates, template categories, or users. The detailed methods of obtaining such sequences will be discussed in the following sections. In this section, for convenience, we will simply use an *item* to refer to a template

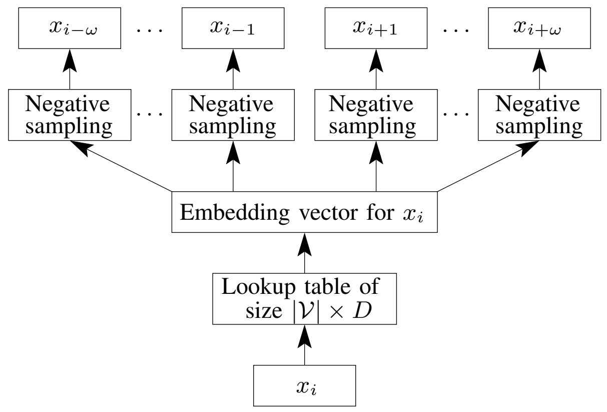


Fig. 1. Embedding learning framework

or a user and denote an item by x. A sequence of length L is denoted by x_1, x_2, …, x_L.

(ii) **Predicting the context**. A key method of the word embedding approach [8] is to use an item to predict its surrounding items, which is referred to as the *context* of the item [7]. The objective for this approach is to maximize the average log probability of the context items given the center item, i.e.,

$$\max \frac{1}{L} \sum_{i=1}^{L} \sum_{-\omega \le j \le \omega, j \ne 0} \log p(x_{i+j}|x_i)\,,$$

where L is the length of a given sequence and ω is the size of the context. Let $\mathcal{V}$ be the vocabulary and D the dimension of the embedding space. Let $\mathbf{M}_x$ be the row vector of matrix $\mathbf{M}$ corresponding to item x. The conditional probability $p(x_{i+j}|x_i)$ can be computed by the following softmax function

$$p(x_{i+j}|x_i) = \frac{\exp\left(\mathbf{W}_{x_{i+j}}\mathbf{E}_{x_i}^T + \mathbf{b}_{x_{i+j}}\right)}{\sum_{k=1}^{|\mathcal{V}|}\exp\left(\mathbf{W}_{x_k}\mathbf{E}_{x_i}^T + \mathbf{b}_{x_k}\right)}$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}|\times D}$ is the weight matrix (where the rows are indexed by items), $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times D}$ is the item embedding matrix, and $\mathbf{b} \in \mathbb{R}^{|\mathcal{V}|\times 1}$ is the bias vector.

(iii) **Negative sampling**. The above formulation to compute the conditional probability $p(x_{i+j}|x_i)$ can be inefficient because the denominator involves all the items in the vocabulary and we need to compute this conditional probability for every learning step, which makes the approach above inapplicable to a large number of sequences. Therefore, we use the technique of negative sampling [8], which is adapted from the Noise Contrastive Estimation (NCE) that presumes an effective model should be able to differentiate true labels from noise [24]. When learning the embedding, negative sampling helps to differentiate the true context items from several randomly sampled negative items when making predictions with the centered item using logistic regression. Negative sampling computes the log probability in the objective function by

$$\log p(x_{i+j}|x_i) = \log\sigma\left(\mathbf{W}_{x_{i+j}}\mathbf{E}_{x_i}^T + \mathbf{b}_{x_{i+j}}\right) + \sum_{n=1}^{K}\mathbb{E}_{x_n\sim P_{\text{neg}}(x)}\left[\log\sigma\left(-\mathbf{W}_{x_n}\mathbf{E}_{x_i}^T - \mathbf{b}_{x_n}\right)\right]$$

where $\sigma(x) = 1/[1+\exp^{-x}]$ is the sigmoid function, K is the number of negative samples, and $P_{\text{neg}}(x)$ is the distribution with which we sample the negative items. Figure 1 illustrates the process of embedding learning. By the end of this process, the embedding matrix $\mathbf{E}$ contains the learned representations for each item.

C. Template Embedding

To generate email template embeddings bearing the desired properties described in Section IV-A, we propose to exploit the collective and common sequential behavior patterns among a large number of users. In particular, we propose to utilize the email *template chains* generated by each user. A template chain is a series of email templates one user receives sequentially. Each template chain (i) reflects the sequential causality among email templates obtained from that user's sequential behaviors and (ii) retains template similarity information obtained from that user's continuous interests. Specifically, the templates e_i^u user u receives form a template chain

$$\mathcal{E}^u = e_1^u \to e_2^u \to \ldots \to e_{|\mathcal{E}^u|}^u\,,$$

where the templates are ordered by their received timestamps, i.e.,

$$t(e_i^u) < t(e_{i+1}^u) \text{ for } i = 1, 2, \ldots, |\mathcal{E}^u| - 1\,.$$

We use the collection of all the sequences $\{\mathcal{E}^u | u \in \mathcal{U}\}$ as input (features) to the embedding learning process described above.

The obtained template representation using such template sequences will have the desired properties described in Section IV-A because: (i) Similar templates often appear in the same contexts, e.g., various types of video games often appear in the context of ordering gaming consoles and purchasing other video games. In the embedding learning process, templates frequently appearing in the same contexts will be projected to positions with close proximity. Therefore, similar templates will be close to each other. In other words, the frequent occurrence of such template-context patterns helps the learning of an effective representation despite of the interleaving of different email causal threads (e.g., order received $\to$ air-ticket receipt $\to$ item shipped $\to$ hotel booking confirmed $\to$ item review request $\to$ dinner reservation). (ii) By utilizing the contexts obtained from template chains, templates frequently appearing in the same context are often sequentially correlated. For example, Car rental and Sightseeing are sequentially correlated, and they often appear in the same context of Hotel booking and Air-tickets purchase. Therefore, sequentially correlated templates will also be close to each other. (iii) Using negative sampling to replace the full softmax function will significantly improve the computation efficiency and hence enable the embedding learning for a large amount of templates.

To learn embeddings of template categories, we can simply replace the templates with their categories in the template sequence.

D. User Embedding

Next, we discuss the approach to obtaining user embeddings. B2C emails are sent from companies to users and companies send the same email templates to users when users have the same requests or interests. Therefore, for email category predictions, it indicates a type of close connection among users when they receive the same email template around the same time. For example, some users may frequently shop online during weekends or like purchasing items that are on-sale (within a limited period) and thus they will receive payment confirmation and item shipping emails around the same time. Some users may enjoy taking vacations during holiday seasons (e.g., Christmas) and hence receive flight check-in and hotel check-out emails at similar times. Based on this observation, we collect together the users who have received and opened the same B2C email templates and organize these users into a sequence where the users are ordered by the time they received the template. Formally, for a template e, the user sequence formed by e is

$$\mathcal{U}^e = u_1^e \to u_2^e \to \ldots \to u_{|\mathcal{U}^e|}^e \,,$$
$$\text{where } \; t(e^{u_i}) \leq t(e^{u_{i+1}}) \; \text{ and } \; o(e^{u_i}) = 1\,.$$

Note that a user may appear multiple times in the sequence if she has received and opened the template multiple times. The learned user embedding using such user sequences as input will have the desired properties because the contexts are formed by users with connections due to similar behaviors or interests and such connections also consider the temporal or sequential information.

Comparing the user sequence and template sequence, we can observe that user sequences are the *inverse* or *transpose* of template sequences in that: (i) the template sequences are formed for individual users while the user sequences are for each template; (ii) the template sequences represent emails *received* by each user while the user sequences are formed by emails *sent* by companies; (iii) the template sequences are created partially due to the causality of users' sequential behavior while the user sequences are in part created by the strategy of companies sending notifications or confirmations. Using both the template and user sequences, the majority of the information from users' email sequences is exploited.

V. Embedding for Email Prediction

In this section, we describe prediction models for email category predictions, with which we show how the learned embeddings can be used in downstream applications.

A. Embedding for Receiving-based Prediction

Recall that given users' previously received emails, the receiving-based email category prediction is to predict the email categories users will receive in the next few days. For this prediction task, an existing work [15] has shown that the long short-term memory (LSTM) produces better prediction performance than the Markov-chain based method. The used LSTM [25] has multiple layers of memory blocks and applies the dropout technique [26] to the connections between different layers. Specifically, let $\mathbf{h}_t^l$ be the vector of a hidden state at layer l at time step t. The transformation from input $\mathbf{h}_t^{l-1}$ and $\mathbf{h}_{t-1}^l$ to output $\mathbf{h}_t^l$ by an LSTM block at layer l is:

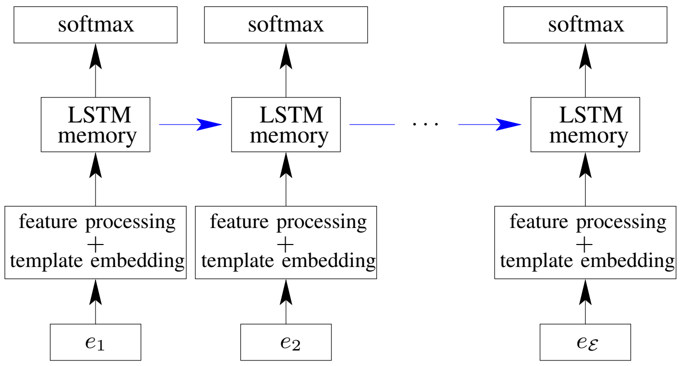

softmax
softmax
softmax
LSTM memory
LSTM memory
LSTM memory
· · ·
feature processing + template embedding
feature processing + template embedding
feature processing + template embedding
e_1
e_2
$e_{\mathcal{E}}$

Fig. 2. LSTM for email category prediction

$$\begin{aligned}
\mathbf{h}_t^d &= \mathcal{D}(\mathbf{h}_t^{l-1}) \\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i \mathbf{h}_t^d + \mathbf{U}_i \mathbf{h}_{t-1}^l + \mathbf{b}_i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f \mathbf{h}_t^d + \mathbf{U}_f \mathbf{h}_{t-1}^l + \mathbf{b}_f\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o \mathbf{h}_t^d + \mathbf{U}_o \mathbf{h}_{t-1}^l + \mathbf{b}_o\right) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c \mathbf{h}_t^d + \mathbf{U}_c \mathbf{h}_{t-1}^l + \mathbf{b}_c\right) \\
\mathbf{h}_t^l &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\mathcal{D}$ is the dropout operator that sets a random subset of the input $\mathbf{h}_t^{l-1}$ to zero, $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the input, forget, and output gate vectors, $\mathbf{c}_t$ is the cell state vector, and $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are the weight matrices and bias vector, respectively. The operators σ and $\tanh$ represent the sigmoid activation function $\sigma(x) = 1/[1 + \exp(-x)]$ and hyperbolic tangent activation function $\tanh(x) = [1-\exp(-2x)]/[1+\exp(-2x)]$ respectively, and $\odot$ represents the Hadamard product (i.e., element-wise multiplication).

The output layer uses an affine transformation and a softmax function to compute the probability distribution over all categories (detailed in the Section V-B). We can train this LSTM with back propagation through time.

The input layer (i.e., features) consists of the template category feature and several temporal features. The category feature is represented as a one-hot encoding vector (i.e., a $|\mathcal{C}|$-dimensional vector where the $c(e)$-th component equals 1 while all others are 0 for a category $c(e) \in \mathcal{C}$). The temporal features are the day of week, period of month, and timestamp gap, which are represented as a 7-dimensional one-hot vector, 3-dimensional one-hot vector, and a real value, respectively.

For this task, we can add the email template embedding as an additional feature, i.e., append the template embedding vector to the input layer. Figure 2 illustrates such a configuration. Since the template embedding contains the information of similarity and sequential correlation among templates, the pre-trained embedding will be an informative feature for this prediction task.

B. Embedding for Interest-based Prediction

Different from the receiving-based prediction, given users' previously *opened* email categories, the interest-based prediction task is to predict the next email category users will open. To the best of our knowledge, there is no existing model designed specifically for this task. Therefore, to explore a different type of network infrastructure, we propose to use a multilayer perceptron (MLP) for this prediction task. MLP is the simplest form of a deep network and has shown to be effective in many applications [27].

Specifically, the MLP we use for the interest-based prediction task is as follows. The connections between layers are fully connected, and the neurons are *rectified linear units* (ReLUs), i.e.,

$$\mathbf{h}_l = \max(\mathbf{W}_{l-1}\mathbf{h}_{l-1} + \mathbf{b}_{l-1}, \mathbf{0}) ,$$

where $\mathbf{h}_l$ is the state of the neurons at layer l and $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector, respectively. Let $\mathbf{h}_o$ be the state of the output layer. We obtain a $|\mathcal{C}|$-dimensional logit vector $\mathbf{z}$ from $\mathbf{h}_o$ with another affine transformation

$$\mathbf{z} = \mathbf{W}_z\mathbf{h}_o + \mathbf{b}_z .$$

The predicted probability $\mathbf{r}$ of being opened by users over the $|\mathcal{C}|$ categories is computed by a softmax function

$$\mathbf{r} = \boldsymbol{\sigma}(\mathbf{z}) , \quad \text{where} \quad \boldsymbol{\sigma}(\mathbf{z})_i = \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{|\mathcal{C}|}\exp(\mathbf{z}_j)} .$$

We also apply the dropout technique at the last hidden layer during training. We use the cross-entropy loss and train this MLP with back propagation.

In the previous receiving-based prediction task, we use the template embedding as an additional feature, to investigate the predictive power and effectiveness of the learned embedding, we use only the embedding vectors as features for this interest-based prediction task. We also use the user embedding as features for this task since the user embedding is obtained from open emails. Figure 3 illustrates the configuration of the prediction model.

In this setting, there are two approaches to using the learned embeddings. One approach is to use the embeddings as features, which means the embedding is fixed during training and the update through back propagation is not applied to the embedding (i.e., input) layer. The other approach is to use the learned embeddings as initializers for the corresponding sparse feature embeddings, which are model parameters and normally apply random initialization (with this approach we can also obtain a new embedding that is adapted to the task at hand). We will investigate the different ways of using the learned embeddings in the next section.

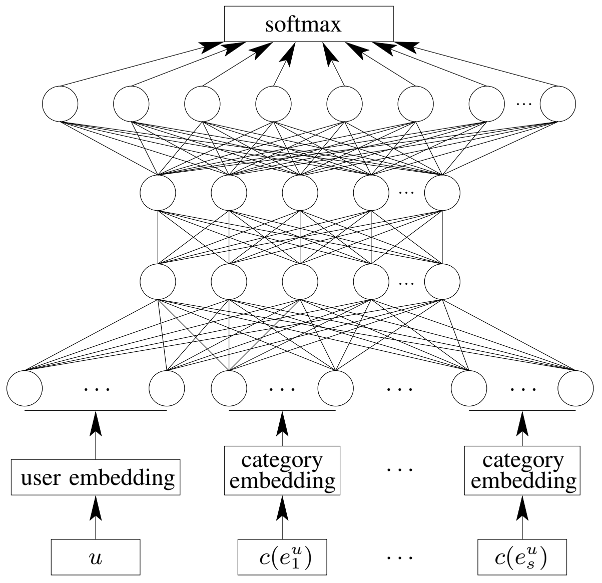


Fig. 3. MLP for email category prediction

VI. Experiments

Data sets. The data set we use for experiments is from Gmail, and consists of data from 273,000 anonymized users and 35 million B2C emails spanning 90 days. Each email has an artificial identifier (ID), an anonymized artificial user ID, a template ID, a category label, a received timestamp, and an indicator of whether the corresponding email was opened or not. All other information and attributes including the entire content of the email are removed to protect users' privacy. The email template is obtained using the techniques introduced in Section II (i.e., by clustering email subjects and matching the fixed components in the same cluster). The category label is from a pre-defined taxonomy used by Gmail.

Embedding learning. For all embedding learning, we use the unigram (item frequency) distribution with the frequency raised to the 3/4rd power as the negative distribution $P_{\text{neg}}(x)$ (since empirical experiments show that this distribution outperforms unigram and uniform distributions [8]). We use the mini-batch stochastic gradient decent method to train the embedding. Unless stated otherwise, the number of negative samples is set to 100, the context size (i.e., number of left or right items to be considered context) is set to 1, and we train the embedding with a batch size of 128 and an initial learning rate of 0.05 with linear decay for 100 epochs. We implement all the algorithms with TensorFlow.

A. Experiment on Receiving-based Prediction

We first conduct experiments for the receiving-based prediction.

Setup. To compare with the existing work [15], we use the email templates of 18 categories including coupon, event, shopping, etc. We split the data into training and testing data at the 45 day midpoint. We use a prediction time window of 3 days and generate about 1.7M training and 490k testing instances with the data of 43k users. We learn the template embedding with template ID sequences and use the learned template embedding as an additional feature. To investigate the effectiveness of an embedding learned with general data, we sample another 64k users and use the template sequences in the training half of (in total) 107k users to learn the template

TABLE I
PERFORMANCE WITH TEMPLATE EMBEDDING

Category	SR@1		MRR	
	LSTM	LSTM+Emb	LSTM	LSTM+Emb
Overall	0.8808	**0.8853**	0.9270	**0.9301**
Coupon	**0.8018**	0.8003	**0.8803**	0.8790
Event	**0.2892**	0.2863	0.5413	**0.5422**
Shopping	0.2692	**0.2800**	0.5071	**0.5125**
Tracking	0.0860	**0.0895**	0.2919	**0.3051**
Newsletter	0.4540	**0.4631**	**0.6556**	0.6519
Flight	0.0428	**0.0649**	0.2114	**0.2345**
Hotel	0.0267	**0.0648**	0.1675	**0.2042**
Restaurant	0.0162	**0.0243**	0.1221	**0.1460**
Car Rental	0.0244	**0.0320**	0.1105	**0.1314**

embedding. The template embedding dimension is set to 200, context size set to 5, and training epochs set to 10. We use an LSTM with one hidden layer and set the dropout ratio to 0.1 since in such setting the LSTM model has the best performance [15]. We train the LSTM with 10 epochs and a batch size of 50.

Metrics. We use the success rate at one (**SR@1**) and mean reciprocal rank (**MRR**) to measure the performance. Let $\mathcal{T}$ be the set of testing instances. Let $c_1^{(\tau)}, c_2^{(\tau)}, \ldots, c_{|\mathcal{C}|}^{(\tau)}$ be the predicted categories in decreasing probability for testing instance $\tau \in \mathcal{T}$, and let $\mathcal{L}^{(\tau)}$ be the set of true categories the user receives within the predictive window. The SR@1 measures the portion of instances where the first predicted category is one of the true categories, i.e.,

$$\text{SR@1} = \frac{|\{\tau \mid c_1^{(\tau)} \in \mathcal{L}^{(\tau)}\}|}{|\mathcal{T}|} .$$

The MRR evaluates the ranking quality of the predicted categories and is computed by the mean of the inverse rank of the first correct prediction, i.e.,

$$\text{MRR} = \frac{1}{|\mathcal{T}|} \sum_{\tau} \frac{1}{\min\{k \mid c_k^{(\tau)} \in \mathcal{L}^{(\tau)}\}} .$$

1) Results: Table I reports the results of the LSTM model (**LSTM**) proposed by [15] and the LSTM model with the additional template embedding feature (**LSTM+Emb**). We can see that overall (for all the 18 categories, shown at the third line of Table I), after adding the template embedding feature, the performances measured by both SR@1 and MRR are improved. This demonstrates that the information carried by the template embedding on the similarity and sequential correlation among templates is indeed helpful for predicting the future email categories.

To investigate the effect on each category, we also train a separate model for each of a few randomly selected categories. The results, in decreasing category frequencies, are listed in the lower half of Table I. We can see that, for high frequency categories such as *Coupon* and *Event*, adding the template embedding even slightly deteriorates the performance. This might be due to the fact that a high frequency category corresponds to many templates, which in turn results in many similar or sequentially correlated templates, and this large amount of templates imposes much noise which distracts the LSTM. For many low frequency categories, however, the performance is significantly improved. Although this may be in part due to the usage of more features, another possible reason is that with the template embedding we can find many similar templates which also have correspondence with the low frequency category, and this indirectly increases the number of low frequency categories. For some categories, e.g., *Flight* and *Hotel*, the improvement is relatively larger than other ones. This may be because these categories have a stronger sequential correlation with other categories and the sequential information carried by the template embedding helps more for their predictions. Many of these low frequency categories are often of higher interest and importance because they often correspond to more important personal activities than close-to-spam advertisements. Predicting the arrival of these low frequency categories is also in general more difficult than those high frequency ones. The fact that using the pre-trained embedding can significantly enhance the prediction for these low frequency categories demonstrates the effectiveness of the proposed techniques. For the *Newsletter* category, although the number of training instances is lower than Event etc., the model's predictive performance for this category is better. This may be because many news letters are sent out periodically and the used temporal features can be quite informative in this respect. For this category, adding the template embedding still improves the performance in terms of SR@1.

B. Experiment on Interest-based Prediction

Next, we evaluate the effectiveness of the learned embedding with the interest-based prediction task.

Setup. For this task, we consider only the emails *opened* by users and use 80% of the opened emails for training, 10% for validation, and 10% for testing. We use the previous 30 opened categories to predict the next 1 category that will be opened and generate 5.6M training, 720k validation, and 680k testing instances. To investigate more fine-grained correlations among email categories, we use more than 1,400 template categories such as Jobs, Football, and Computer Components. We obtain template category sequences only with the training data and set the dimension of template category embedding to be 32. Unless stated otherwise, by default, we use an MLP with 3 hidden layers with all the hidden layer sizes set to 512 and use a dropout ratio of 0.1. We train the MLP with the Adagrad stochastic gradient descent [28] with an initial learning rate of 0.005 and a batch size of 4096 for 150,000 steps (approximately 110 epochs).

Metrics. We measure the performance by accuracy (**accuracy**) and recall at 5 (**recall@5**). Let $\zeta^{(\tau)}$ be the true category that the user opens next for a testing instance $\tau \in \mathcal{T}$. The accuracy measures the portion of correct predictions and is computed by

$$\text{Accuracy} = \frac{|\{\tau \mid c_1^{(\tau)} = \zeta^{(\tau)}\}|}{|\mathcal{T}|} .$$

TABLE II
NEAREST NEIGHBORS OF CATEGORY EMBEDDING

Email category	Top-3 nearest neighbors in the embedding space		
Travel	Hotel	Travel Insurance	Airport Parking
Open Source	Scripting Language	Data Management	Machine learning
Weight Loss	Nutrition	Low Carbohydrate Diet	Fitness
Babies	Baby Care	Baby Feeding	Pregnancy
Career Planning	Internship	Jobs	Recruitment
Basketball	Sports Training	Sports News	Football
Computer Hardware	Monitors	Flash Drives	Computer Components

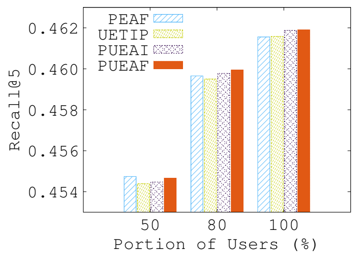
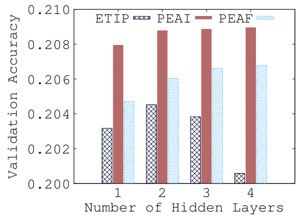
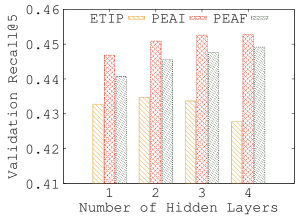


Fig. 4. Cold-start situations

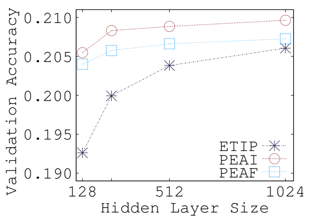


(a) Validation accuracy

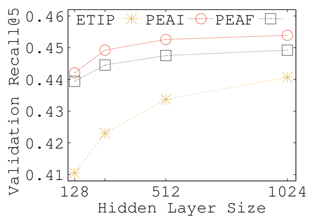


(b) Validation recall@5

Fig. 5. Effect of the number of hidden layers



(a) Validation accuracy



(b) Validation recall@5

Fig. 6. Effect of hidden layer sizes

The recall@5 evaluates the predictions at the first five positions and equals

$$\text{Recall@5} = \frac{|\{\tau \,|\, \zeta^{(\tau)} \in \{c_i^{(\tau)} \,|\, i = 1, \ldots, 5\}\}|}{|\mathcal{T}|}.$$

Compared methods. We compare the methods of (i) using a sparse feature *e*mbedding *t*rained *i*n the training *p*rocess (**ETIP**), (ii) using the *p*re-trained (learned) *e*mbedding *a*s an *i*nitializer for the sparse feature embedding (**PEAI**), and (iii) using the *p*re-trained *e*mbedding *a*s fixed *f*eatures (**PEAF**), as discussed in Section V-B.

1) Results: We first investigate the category embedding in the embedding space. Table II shows case studies for a few randomly selected categories, where we list their top-3 nearest neighbors in the embedding space. We can see that the results are quite interesting and the category embedding does present the desired properties described in Section IV-A, which require that similar and sequentially correlated items are close to each other. For example, for users interested in *Weight Loss*, they may also be interested in *Nutrition* and *Low Carbohydrate Diets* and they may later or have already registered as a *Fitness* gym member. With the learned embedding, we can also discover interesting patterns, e.g., the sequential correlation between *Travel Insurance* and *Airport Parking*.

We then study the effect of hyper-parameters of MLP. Figures 5 and 6 show the validation performances of different methods when we vary the number and size of hidden layers, respectively. From Figure 5, we can see that the performances, measured by both accuracy and recall@5, of ETIP first increase and then decrease with the increase of the number of hidden layers, while the performances of PEAI and PEAF keep increasing with a slower increase rate. We can also see from Figure 6 that with the increase of hidden layer sizes the accuracy and recall@5 of all the three methods increase with a slower increase rate. These experiments indicate that it is a valid choice to set the number and size of hidden layers to be 3 and 512, respectively. We can also see that, under all these different settings, the relative validation performances of the three methods remain consistent, with PEAI and ETIP having the best and worst performances respectively and PEAF lying in between.

Figure 7 plots the training loss of different methods against the training steps. We can see that the training loss of PEAI and PEAF is on average smaller that of ETIP. This indicates that using the pre-trained embedding, either as an initializer or features, can help obtain a model that fits the training data better. We can also observe that PEAI and PEAF achieve convergence much faster than ETIP, which indicates that using the pre-trained embedding can significantly reduce the training time.

Figures 8(a), 8(b), and 8(c) present the validation loss, accuracy, and recall@5 against the training steps, respectively. We can observe that, by all evaluation metrics, PEAI and PEAF consistently outperform ETIP, with PEAI slightly outperforming PEAF. As the training process goes on, the validation performances of ETIP become closer to those of PEAI and PEAF, but are still inferior to PEAI and PEAF. This again demonstrates that using the pre-trained embedding can help obtain a better model in a shorter time.

Table III reports the accuracy and recall@5 of different methods on the testing set. The Overall column shows the results on all categories. The remaining columns show the testing results on a few randomly selected categories in descending frequency (from left to right). We can see that, overall, PEAI has the highest accuracy and recall@5, ETIP has the lowest, and PEAF's performances lie in between. This

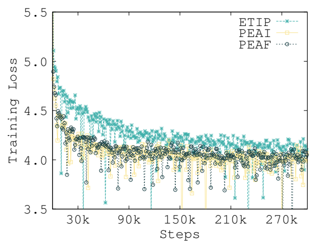
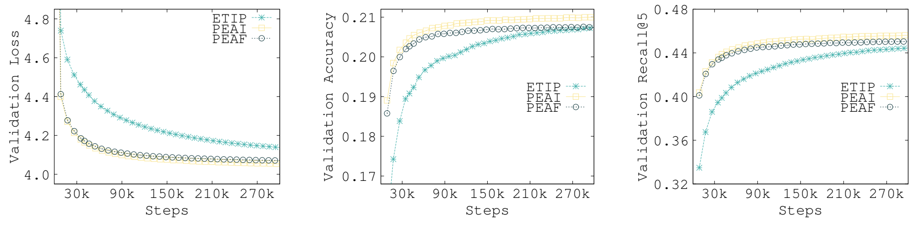


Fig. 7. Training loss



(a) Validation loss (b) Validation accuracy (c) Validation recall@5

Fig. 8. Validation performance

TABLE III
TESTING PERFORMANCE FOR TEMPLATE CATEGORY EMBEDDING

Category	Overall		Coupon		Hotel		Restaurant		Basketball	
Model	Accuracy	Recall@5	Accuracy	Recall@5	Accuracy	Recall@5	Accuracy	Recall@5	Accuracy	Recall@5
ETIP	0.20509	0.43456	**0.54333**	0.87637	**0.29346**	0.63730	0.04234	0.09830	0.0	0.00763
PEAI	**0.21074**	**0.45350**	0.53278	**0.87754**	0.27280	**0.63787**	0.05280	0.12287	0.01527	**0.04580**
PEAF	0.20831	0.44809	0.52302	0.87175	0.27343	0.62051	**0.05328**	**0.12457**	**0.01781**	0.03562

reveals that using the pre-trained embedding as the initializer for the sparse featured embedding can help obtain a much better model. This also signals that the pre-trained embedding carries helpful information that can not be obtained from the training instances. Even making the pre-trained embedding fixed as features can still provide more information than a specifically trained sparse feature embedding for this task.

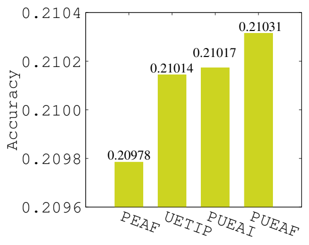
Inspecting from individual categories, for high frequency categories such as *Coupon* and *Hotel*, the accuracy of ETIP is better than the other two methods using the pre-trained embedding, but they all have similar recall@5. This indicates that the model that uses a freshly trained embedding tends to rank those high frequency categories highest, while the models using the pre-trained embedding tend to rank them slightly lower but still within top-5. For low frequency categories such as *Restaurant* and *Basketball*, as with the receiving-based prediction task, using the pre-trained embedding helps improve the model's performances on these low frequency categories, and making the pre-trained embedding fixed as features gives slightly better performances. This phenomenon indicates that letting the embedding be free (trainable) parameters during training will make the model favor high frequency categories. Using the pre-trained embedding as an initializer for the trainable embedding achieves a good trade-off between the high frequency and low frequency categories.

2) User Embedding: We also investigate the effect of using the user embedding for this task.

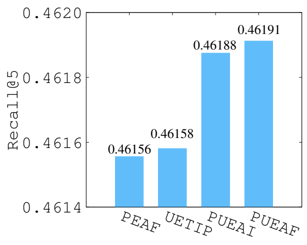
Setup. We split each user's templates by time and use the first 70% for training and the remaining for testing. As a result, we generate 4.4M training and 1.6M testing instances. We set the dimension of user embedding to 64 and learn a new category embedding with the training data. All other parameters for embedding generation and MLP training remain the same. To focus on the effect of user embedding, we make the category embedding fixed as features. We compare the methods of (i) PEAF, (ii) PEAF with *u*ser *e*mbedding *t*rained *i*n the training *p*rocess (**UETIP**), (iii) PEAF with *p*re-trained *u*ser *e*mbedding *a*s *i*nitializer (**PUEAI**), and (iv) PEAF with *p*re-trained *u*ser *e*mbedding *a*s fixed *f*eature (**PUEAF**). All the models in the remaining experiments are repeatedly trained and tested three times and the average results are reported.

Figure 9 plots the accuracy and recall@5 of different methods. We can see that after adding the user embedding, compared with PEAF, the model performances are further improved. Given that there are more than 1.5 million testing instances, the improvement, albeit small, is statistically meaningful (we will further investigate this in the next set of experiments). We can also see that using the pre-trained user embedding gives better performances than training a fresh user embedding especially in terms of recall@5. Since the previous experiments have shown that training a fresh embedding tends to favor more frequent items, UETIP will also favor power users with many opened templates and hence more training instances. This essentially makes the model resemble the behavior of power users and hence less diverse, which decreases the recall@5. That the performance of PUEAF is better than PUEAI indicates that the distribution of the number of opened emails (i.e., training instances) across users is less skewed (i.e, more even) than that of the category frequencies, which leads to that it is less beneficial to make the model favor power users. Overall, using the pre-trained user embedding can help improve the performance.

In the last set of experiments, we study whether the user embedding will help *cold-start* users, i.e., users whose email open patterns are not seen by the MLP during training. We simulate the cold-start situation by (i) training the MLP with only a portion of users' data (user embedding is still trained with all users' data) and (ii) testing the performance of MLP with all users' data (i.e., all 1.6M testing instances). Figure 4 presents the recall@5 of different methods when the portion of users' data used for MLP training is varied from 50% to



(a) Testing accuracy



(b) Testing recall@5

Fig. 9. Performance with user embedding

100%. We can see that when we use only half of the users' data for training, the performances of all the models using user embedding are even worse than that of PEAF which does not use any user related features. This indicates that users' email opening behavior is quite diverse since 50% of the users cannot represent the majority behavioral patterns, and utilizing user similarity at this level will distract the MLP. The situation is changed when we use 80% of the users for training. The performances of PUEAI and PUEAF are better than that of PEAF, which means considering personalization and user similarity at this level is beneficial and the pre-trained embedding can help the prediction of both seen and unseen users. The performance of UETIP is still worse than PEAF, indicating using a freshly trained user embedding is not a good choice at this level. Another interesting phenomenon is that although using 100% of the data for training can achieve a better performance than using only 50%, the performance measured by recall@5 only increases from around 0.455 to around 0.462, which in turn indicates that the improvement by using user embedding, albeit small, is statistically significant.

VII. Conclusions

We have proposed a simple yet efficient and effective framework for learning embeddings of templates, template categories, and users from email data. The framework only requires obtaining item sequences with which we identify item contexts. Using only a single-layer neural network and the negative sampling technique, the embedding can be trained with large-scale data in a short time and can carry a large amount of desired information. For either the receiving-based email category prediction with LSTM or the interest-based prediction with MLP, the pre-trained embedding can be effectively utilized and can be conveniently used as either fixed features or embedding initializers. Extensive experiments with thousands of users and millions of emails have demonstrated that with the learned embedding, we can find clusters of templates with interesting correlations or users with similar interests. The experiments have also shown that with the pre-trained embedding the predictive performances of LSTM and MLP for the two email category prediction tasks can be significantly improved and that the pre-trained embedding can be used to address cold-start problems. We hope that the simple and efficient framework and interesting results can attract more attention to email intelligence.

References

[1] N. Ailon, Z. S. Karnin, E. Liberty, and Y. Maarek, "Threading machine generated email," in *WSDM*, 2013, pp. 405–414.

[2] N. Avigdor-Elgrabli, M. Cwalinski, D. Di Castro, I. Gamzu, I. Grabovitch-Zuyev, L. Lewin-Eytan, and Y. Maarek, "Structural clustering of machine-generated mail," in *CIKM*, 2016, pp. 217–226.

[3] Y. Maarek, "Web mail is not dead!: It's just not human anymore," in *WWW*, 2017, pp. 5–5.

[4] D. Di Castro, Z. Karnin, L. Lewin-Eytan, and Y. Maarek, "You've got mail, and here is what you could do with it!: Analyzing and predicting actions on email messages," in *WSDM*, 2016, pp. 307–316.

[5] Y. Maarek, "Is mail the next frontier in search and data mining?" in *WSDM*, 2016, pp. 203–203.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[7] M. Rudolph, F. Ruiz, S. Mandt, and D. Blei, "Exponential family embeddings," in *NIPS*, 2016, pp. 478–486.

[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[9] J. Proskurnia, M.-A. Cartright, L. Garcia-Pueyo, I. Krka, J. B. Wendt, T. Kaufmann, and B. Miklos, "Template induction over unstructured email corpora," in *WWW*, 2017, pp. 1521–1530.

[10] J. B. Wendt, M. Bendersky, L. Garcia-Pueyo, V. Josifovski, B. Miklos, I. Krka, A. Saikia, J. Yang, M.-A. Cartright, and S. Ravi, "Hierarchical label propagation and discovery for machine generated email," in *WSDM*, 2016, pp. 317–326.

[11] N. Potti, J. B. Wendt, Q. Zhao, S. Tata, and M. Najork, "Hidden in plain sight: Classifying emails using embedded image contents," in *WWW*, 2018, pp. 1865–1874.

[12] Y. Sheng, S. Tata, J. B. Wendt, J. Xie, Q. Zhao, and M. Najork, "Anatomy of a privacy-safe large-scale information extraction system over email," in *KDD*, 2018, pp. 734–743.

[13] M. Bendersky, X. Wang, D. Metzler, and M. Najork, "Learning from user interactions in personal search via attribute parameterization," in *WSDM*, 2017, pp. 791–799.

[14] I. Gamzu, Z. Karnin, Y. Maarek, and D. Wajc, "You will get mail! predicting the arrival of future email," in *WWW*, 2015, pp. 1327–1332.

[15] A. Zhang, L. Garcia-Pueyo, J. B. Wendt, M. Najork, and A. Broder, "Email category prediction," in *WWW Companion*, 2017, pp. 495–503.

[16] J. L. Elman, "Distributed representations, simple recurrent networks, and grammatical structure," *Machine learning*, vol. 7, no. 2-3, pp. 195–225, 1991.

[17] G. E. Hinton, "Distributed representations," 1984.

[18] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[19] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[21] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *arXiv preprint arXiv:1705.02801*, 2017.

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[23] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015, pp. 3156–3164.

[24] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 307–361, 2012.

[25] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[26] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[28] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.