Toward Activity Discovery in the Personal Web

Tara Safavi
University of Michigan*
tsafavi@umich.edu

Adam Fourney
Microsoft
adamfo@microsoft.com

Robert Sim
Microsoft
rsim@microsoft.com

Marcin Juraszek
Microsoft
marcinj@microsoft.com

Shane Williams
Microsoft
shanewil@microsoft.com

Ned Friend
Microsoft
Ned.Friend@microsoft.com

Danai Koutra
University of Michigan
dkoutra@umich.edu

Paul N. Bennett
Microsoft
pauben@microsoft.com

ABSTRACT

Individuals' personal information collections (their emails, files, appointments, web searches, contacts, etc) offer a wealth of insights into the organization and structure of their everyday lives. In this paper we address the task of learning representations of personal information items to capture individuals' ongoing activities, such as projects and tasks: Such representations can be used in activity-centric applications like personal assistants, email clients, and productivity tools to help people better manage their data and time. We propose a graph-based approach that leverages the inherent interconnected structure of personal information collections, and derive efficient, exact techniques to incrementally update representations as new data arrive. We demonstrate the strengths of our graph-based representations against competitive baselines in a novel intrinsic rating task and an extrinsic recommendation task.

ACM Reference Format:
Tara Safavi, Adam Fourney, Robert Sim, Marcin Juraszek, Shane Williams, Ned Friend, Danai Koutra, and Paul N. Bennett. 2020. Toward Activity Discovery in the Personal Web. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371828

1 INTRODUCTION

In his influential 1945 essay *As We May Think*, Vannevar Bush outlined his vision for a "memex", a device in which "an individual stores all his books, records, and communications, and...may be consulted with exceeding speed and flexibility" [4]. His vision is well-known, having inspired generations of research and development in computing [9, 30]. Perhaps less known, however, is that Bush motivated his memex by the *associative* nature of the human mind: "With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts...[it] is awe-inspiring beyond all else in nature". In this work we expand on the idea of *associating individuals' personal information items*—what Bush called books, records, and communications, and what today might be files, emails, and messages—according to their higher-level purposes and usages. Specifically, our central research question is: **Can we learn representations of personal information objects (files, emails, messages, appointments, search queries, contacts, etc) to support downstream settings that rely on knowledge of people's activities (projects, hobbies, tasks)?**

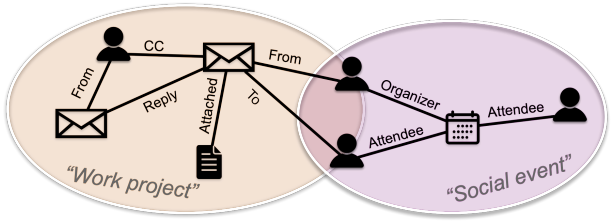


Figure 1: A personal web consisting of two activities.

Motivations and challenges. The goal of activity discovery is to help people better organize, retrieve, and utilize their personal information. For example, modern email clients support tagging and foldering, but individuals struggle to maintain these efforts because manual curation is costly [12, 13, 33, 34]. We envision next-generation email clients that automatically learn users' ongoing activities to organize or even prioritize emails and meetings based on those activities. Semantic and conversational search systems can also benefit from inferring users' activities. For example, such systems could allow users to directly search by concept or activity (e.g., "Show me all receipts related to my home remodel"). Even helping people understand how they spend their time by activity can be useful for productivity-related reflection and planning.

However, this direction comes with unique challenges. For one, people's activities are complex and fluid. They can exist on varying time scales and evolve over time. Some activities overlap with, or subsume, one another. Ideally, automated approaches to personal

*Work performed while an intern at Microsoft.

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371828

activity discovery should capture such complexity. Another challenge is that of evaluation, which is difficult due to (well-founded) privacy concerns, a lack of standardized methodology, and the high cost of obtaining explicit feedback [35].

This work. Guided by the concept of *associations between items*, we learn representations of people's personal information objects such that objects related by activity have similar representations and can be directly compared regardless of type. We leverage the inherent *graph structure* of personal information collections, augmenting items with interaction- and structure-based links to form what we call a *personal web*. Key contributions of this work include:

- **Methodology**. We propose to model individuals' personal information collections as graphs, and learn unsupervised entity representations with a propagation-based objective. We **derive efficient, exact techniques to update representations as new data arrive**, up to 470× faster than learning from scratch (§ 3, § 6). Our model can **produce human-interpretable representations**, and can also implicitly capture semantic differences in entity types while still representing items in a common space.
- **Evaluation**. We conduct a pair of complementary evaluations. Our first task, which directly tests for activity-specific relationships among individuals' personal information objects (§ 4), is to the best of our knowledge **the first intrinsic evaluation of graph representations involving pairwise human judgments**. By contrast, our second evaluation task, email recipient recommendation, is extrinsic (§ 5). In both tasks, our representations **outperform competitive baselines across a variety of metrics**. We provide in-depth analysis of our evaluation tasks to inform future work in this area.

2 RELATED WORK

Personal information management. Personal information management (PIM) addresses how people organize and retrieve items from their personal information collections [30]; for brevity, we refer the reader to [20] for a comprehensive overview of the literature. A representative, highly influential example is *Stuff I've Seen* (SIS) [9], a unified search index aggregating heterogeneous personal information objects from users' desktops. SIS was followed by *Implicit Query* [11], an email plug-in that automatically displays desktop items related to the user's current email of focus. These early works motivate our goal of unifying heterogeneous personal information objects, as they provide empirical evidence that people prefer such unified systems [9]. Recent works develop privacy-preserving machine learning approaches for PIM, most often for email [1, 35, 36]. For example, Zhao et al. [36] devise CAPERS, a personal recommender system that displays workplace emails to users based on their upcoming meetings. The authors find that an online learning variant of CAPERS performs best with users, which directly motivates our goal of online model updating (§ 3.1).

Activity inference. The literature on inferring individuals' activities over personal information items focuses mostly on email [7, 8, 25]. Dredze et al. [7] devise a supervised method to classify emails into activities by calculating overlap statistics among email messages and known activities. More recently, Qadir et al. [25] learn activities from workplace email with an unsupervised generative model that considers contexts like subject line, recipients, and linguistic features of emails. A major difference between these works and our own is that we wish to discover activities that span multiple *types* of objects, including but not limited to email. From a problem setting perspective, the *TaskTracer* system [26] for activity management over heterogeneous desktop items is more related. However, *TaskTracer* is supervised and requires users to enforce a hierarchy over their data, whereas we take an unsupervised approach.

Graph similarity search. Because we model personal information collections as graphs, our task can be cast as a similarity search problem among nodes in a graph [17, 28]. Similar to our setting from an individual perspective, McAuley and Leskovec [22] learn "social circles" in personal social networks with a generative model that accounts for both links within circles and user-user profile similarity. More recently, vector representations, and in particular dense embeddings of nodes, edges, and/or whole graphs, have become popular as versatile catch-alls for network-based tasks that rely on similar representations of similar objects [14, 16, 18, 24, 29]. While recent work proposes embeddings for professional email networks [19], we are not aware of any approaches specifically tailored to individuals' heterogeneous personal information items.

3 METHODOLOGY

In this section we first define the problem of activity discovery. We then introduce our graph-based representations of heterogeneous personal information items, and show that we can efficiently obtain these representations in both offline and online settings.

3.1 Problem statement

According to Dredze et al. [7], activities are collaborative practices that have state and a goal (e.g., *organizing a conference*). Using this definition, we state the following desiderata for activity discovery over heterogeneous personal information collections:

- **Privacy-preserving**. Personal information collections are highly private and contextual, with overloaded terms and concepts that often do not generalize across individuals [1]. Our model should therefore operate on a per-person basis, directly on personal devices, without leveraging collective patterns across users.
- **Requires no supervision**. Manually organizing personal data into, e.g., social circles or email folders, requires a nontrivial amount of effort [22, 33]. Accordingly, we do not assume the presence of labeled data, although an ideal model should be able to incorporate activity labels if available.
- **Online updates**. Individuals' activities naturally change over time. To handle this evolution, we should be able to incrementally and efficiently update our model as new data arrive.

We state our research problem as follows: **Given an entity e from an individual's heterogeneous personal information collection C, find entities e' that are "related" or "associated" to e in the context of u's activities in a manner that (1) preserves privacy by being learned on an individual, on-device basis, (2) is unsupervised, and (3) can operate in an online setting**.

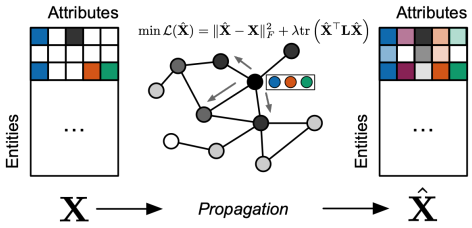
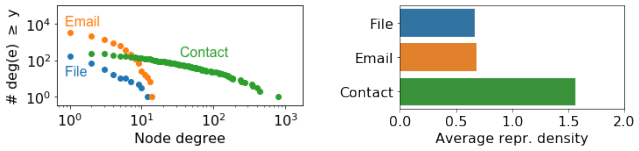


Figure 2: Obtaining representations of personal information items via propagation over the personal web G.



(a) Log-log cumulative degree distribution in the personal web, stratified by type.

(b) Average representation density, using noun phrases as attributes, stratified by type.

Figure 3: Capturing semantic differences in entity types through representation density for a random *Medium* inbox from the Avocado dataset (Table 3). On average, *Contact* nodes have the highest degrees and thus the densest representations, reflecting that people usually participate in more activities than other types of entities.

3.2 Model overview

As our problem statement is open-ended, several classes of methodology could be employed. For example, a retrieval approach would rank the most related entities e' to a query entity e in terms of activities, whereas a clustering approach would directly group entities into activities. To handle this open-endedness we propose to learn versatile representations of heterogeneous personal information items that can be used in numerous settings like retrieval, clustering, and classification. Our approach emphasizes two ideas: First, that personal information objects have inherent graph structure (Figure 1); and second, that closely-connected objects sharing features indicative of activities should have similar representations.

3.2.1 Graph-based representations. Given an individual's personal information collection C, we construct a graph G from C.[1] We call G the **personal web** associated with that individual. Each entity (node) in G has an associated type, such as *Email*, *Calendar Appointment*, or *Contact*, and may be associated with additional temporal and textual features, for example email sent times, subject lines, etc. Each edge in the graph encodes a semantically meaningful relationship between entities. For example, an edge connecting a *Calendar Appointment* to a *Contact* might signify that the appointment was organized or attended by that person. Following standard practice [22], a personal web does not include the individual who owns the data. In practice, a personal web can be instantiated in many different ways, the effects of which we illustrate in § 3.2.3.

We learn entity representations with a propagation process over G that yields similar representations for entities that are closely-connected in G and/or share similar features. The propagation starts from a set of seed entities in G that are associated with "activity-specific" attributes, which are textual, temporal, or other attributes indicative of activities (examples given in § 3.2.2). Formally, let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a matrix in which nonnegative entry X_{ij} corresponds to entity i's membership strength in activity-indicating attribute j. We propagate the seeds' attribute membership strengths X_{ij} across the graph to learn $\hat{\mathbf{X}} \in \mathbb{R}^{n \times p}$, the entity-attribute membership matrix for all entities in G, by minimizing the loss function

$$\mathcal{L}(\hat{\mathbf{X}}) = \left\|\hat{\mathbf{X}} - \mathbf{X}\right\|_F^2 + \lambda \,\mathrm{tr}\left(\hat{\mathbf{X}}^\top \mathbf{L}\hat{\mathbf{X}}\right), \tag{1}$$

where the graph's Laplacian is given as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ for diagonal degree matrix $\mathbf{D}$ and symmetric adjacency matrix $\mathbf{A}$. The first term in (1) constrains the learned attribute values for seed entities to be close to their initial values. The second term enforces graph smoothness and yields similar learned attribute-value distributions for linked entities, controlled by regularization hyperparameter λ (explored in § 5.3). Note that (1) recalls label propagation and convolutions over graphs [21, 37], but such techniques are typically used to predict node classes, which is not our goal. Rather, we use propagation to obtain entity representations: **The i-th entity's final representation is given as $\hat{\mathbf{x}}^i$, the i-th row of $\hat{\mathbf{X}}$.**

3.2.2 Entity attributes. The propagation seed matrix $\mathbf{X}$ maps select entities to attributes that directly or indirectly indicate activities. In the unsupervised case, which is the focus of this work, we consider text-bearing entities like emails and files as seeds, using textual cues to indicate activities. Specifically, we experiment with noun phrase frequencies and latent topic memberships (LSA [6] and LDA [3]) as attributes and their associated strengths: Noun phrases often directly correspond to project, task, or goal names [2], whereas latent topics capture semantic relatedness among groups of documents. Importantly, when noun phrases are used, our approach can produce fully human-interpretable representations, since the columns in $\mathbf{X}$ correspond to natural language. It also naturally handles semi-supervision: We can consider activity labels as attributes if available, although we leave this direction for future work.

3.2.3 Entity type semantics. Our representation approach implicitly captures entity type semantics via *representation density*. Figure 3 illustrates this effect with a graph from the Avocado dataset (§ 5) consisting of the following relations: *Contact-Email*, signifying who sent or received a specific email; *Email-Email*, signifying direct replies on email threads; and *File-Email*, signifying files attached to emails. Figure 3a demonstrates that a node's degree corresponds strongly with its type: The degrees of *Contact* entities are up to orders of magnitude larger than those of *Email*s and *File*s.

Figure 3b demonstrates that *Contact* entity representations are on average twice as dense as *Email* and *File* entity representations, due to their having higher degree. In this case, the higher density of *Contact* entities reflects the idea that people usually participate in more activities than do single emails or files. This effect is an advantage of our approach: We allow entities' representation densities to vary according to their type semantics, while still representing all heterogeneous objects in a shared vector space.

[1] In this paper we experiment with unweighted and undirected graphs, but our model can be straightforwardly extended to handle edge weights and directionality.

3.3 Model inference

3.3.1 Offline setting. We are given a graph G and its corresponding Laplacian matrix $\mathbf{L}$. To find the entity-attribute membership matrix $\hat{\mathbf{X}}$, we first take the derivative of (1) with respect to $\hat{\mathbf{X}}$:

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{X}}} = 2\left(\hat{\mathbf{X}} + \lambda \mathbf{L}\hat{\mathbf{X}} - \mathbf{X}\right).$$

Setting the derivative to 0 and solving for $\hat{\mathbf{X}}$, we obtain

$$\hat{\mathbf{X}} = (\mathbf{I} + \lambda \mathbf{L})^{-1}\,\mathbf{X}, \tag{2}$$

where $\mathbf{I}$ is the identity matrix. In practice, to avoid a computationally prohibitive matrix inversion, we solve for each column of $\hat{\mathbf{X}}$ with Jacobi iteration, which is guaranteed to converge because the matrix inverted in (2) is diagonally dominant:

$$\hat{\mathbf{x}}_i^{(j+1)} = (\mathbf{I} + \lambda \mathbf{D})^{-1}\left(\mathbf{x}_i + \lambda \mathbf{A}\hat{\mathbf{x}}_i^{(j)}\right), \tag{3}$$

where $\mathbf{x}_i$ is the i-th column of $\mathbf{X}$, and $\hat{\mathbf{x}}_i^{(j)}$ is the i-th column of $\hat{\mathbf{X}}$ in the j-th iteration.

3.3.2 Online setting. In a more realistic setting, we wish to incrementally update entity representations $\hat{\mathbf{X}}$ as new data arrive over time. We make the mild assumption that as new data arrive, entity attributes and their strengths can be obtained on-the-fly rather than requiring a full pass over the data; for example, this is true when we take noun-phrases as attributes, but not when we require a decomposition of the full document-term matrix (e.g., LSA, LDA).

The online setting consists of two cases:

- **Case 1**: Only the graph structure changes, meaning a new personal information object arrives (e.g., an email) and/or a new link between objects arrives (e.g., an email is forwarded to a colleague), without observing new attributes.
- **Case 2**: Both the graph structure and the graph's attributes change, meaning in addition to structural changes in G we observe new textual information, like an unseen noun phrase.

We begin with **Case 1**. Given newly observed edge (i, j), let $\Delta\mathbf{D}$ and $\Delta\mathbf{A}$ represent rank-one updates to $\mathbf{D}$ and $\mathbf{A}$, respectively, such that the updated degree matrix is $\mathbf{D}_{\text{new}} = \mathbf{D} + \Delta\mathbf{D}$, the updated adjacency matrix is $\mathbf{A}_{\text{new}} = \mathbf{A} + \Delta\mathbf{A}$, and the updated Laplacian is

$$\mathbf{L}_{\text{new}} = \mathbf{D}_{\text{new}} - \mathbf{A}_{\text{new}} = (\mathbf{D} - \mathbf{A}) + (\Delta\mathbf{D} - \Delta\mathbf{A}) = \mathbf{L} + \Delta\mathbf{L}. \tag{4}$$

Because only the graph structure changes, the online objective matches (1), except that $\mathbf{L}_{\text{new}}$ replaces $\mathbf{L}$. Therefore, following (2), the closed-form solution of the online objective is

$$\hat{\mathbf{X}}_{\text{new}} = (\mathbf{I} + \lambda \mathbf{L}_{\text{new}})^{-1}\,\mathbf{X} = (\mathbf{I} + \lambda \mathbf{L} + \lambda\Delta\mathbf{L})^{-1}\,\mathbf{X}. \tag{5}$$

Equation (5) can be naively solved with Jacobi iteration as in § 3.3.1, but we devise a faster approach that reuses previous computation *and* arrives at the same solution as our offline model. The key to efficiency is that both $\Delta\mathbf{D}$ and $\Delta\mathbf{A}$ are rank one and can be expressed as outer products. Letting $\mathbf{e}_i \in \mathbb{R}^n$ be an indicator vector with its i-th entry equal to one, and zero elsewhere, we can express $\Delta\mathbf{D} = \mathbf{e}_i\mathbf{e}_i^\top + \mathbf{e}_j\mathbf{e}_j^\top$ and $\Delta\mathbf{A} = \mathbf{e}_i\mathbf{e}_j^\top + \mathbf{e}_j\mathbf{e}_i^\top$. It then follows that

$$\begin{aligned}\Delta\mathbf{L} &= \Delta\mathbf{D} - \Delta\mathbf{A} \qquad \rightarrow \text{ From (4)}\\ &= (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top\end{aligned} \tag{6}$$

is also rank one. Using the Sherman-Morrison formula for rank-one updates to a matrix inverse [27], it follows that

$$\begin{aligned}\hat{\mathbf{X}}_{\text{new}} &= (\mathbf{I} + \lambda\mathbf{L} + \lambda\Delta\mathbf{L})^{-1}\mathbf{X} \qquad \rightarrow \text{ From (5)}\\ &= \left(\mathbf{I} + \lambda\mathbf{L} + \lambda(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top\right)^{-1}\mathbf{X} \qquad \rightarrow \text{ From (6)}\\ &= \left((\mathbf{I} + \lambda\mathbf{L})^{-1} - \frac{(\mathbf{I} + \lambda\mathbf{L})^{-1}\lambda(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top(\mathbf{I} + \lambda\mathbf{L})^{-1}}{1 + (\mathbf{e}_i - \mathbf{e}_j)^\top(\mathbf{I} + \lambda\mathbf{L})^{-1}\lambda(\mathbf{e}_i - \mathbf{e}_j)}\right)\mathbf{X}\\ &= \hat{\mathbf{X}} - \frac{(\mathbf{I} + \lambda\mathbf{L})^{-1}\lambda(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top\hat{\mathbf{X}}}{1 + (\mathbf{e}_i - \mathbf{e}_j)^\top(\mathbf{I} + \lambda\mathbf{L})^{-1}\lambda(\mathbf{e}_i - \mathbf{e}_j)},\end{aligned}$$

where on the last line we substitute $(\mathbf{I} + \lambda\mathbf{L})^{-1}\mathbf{X} = \hat{\mathbf{X}}$ as per the closed form in (2). Now, letting

$$\mathbf{u} = (\mathbf{I} + \lambda\mathbf{L})^{-1}\lambda(\mathbf{e}_i - \mathbf{e}_j) \tag{7}$$

and

$$\mathbf{v}^\top = \frac{(\mathbf{e}_i - \mathbf{e}_j)^\top\hat{\mathbf{X}}}{1 + (\mathbf{e}_i - \mathbf{e}_j)^\top\mathbf{u}}, \tag{8}$$

we can write the rank-one update to $\hat{\mathbf{X}}$ as

$$\Delta\hat{\mathbf{X}} = \hat{\mathbf{X}}_{\text{new}} - \hat{\mathbf{X}} = -\mathbf{u}\mathbf{v}^\top, \tag{9}$$

meaning that per newly observed edge, we only need to update $\hat{\mathbf{X}}$ with the outer product in (9).

In **Case 2**, where the new edge (i, j) contains at least one entity with previously unobserved attributes, we propagate these new attributes across the graph using (3) and add the result, along with the result of (9), to $\hat{\mathbf{X}}$. We demonstrate theoretically (§ 3.3.3) and empirically (§ 6.2) that this outer product formulation results in orders of magnitude performance improvement.

3.3.3 Complexity analysis. Let n be the number of entities and m be the number of edges in G. In the offline setting, solving for each column of $\hat{\mathbf{X}}$ requires $O(m)$ time for a fixed number of Jacobi iterations, since the matrix to be inverted in (2) has $O(m)$ nonzero entries. Thus, with p attributes, the total complexity is $O(mp)$.

In the online setting, solving for $\mathbf{u}$ takes $O(m)$ time, again using the Jacobi method. Computing $\mathbf{v}$ takes $O(np)$ time for $\hat{\mathbf{X}} \in \mathbb{R}^{n\times p}$, and taking the outer product $\mathbf{u}\mathbf{v}^\top$ is also $O(np)$ for $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^p$. Therefore, the total complexity of evaluating (9) without observing a new entity is $O(m + np)$. If a new entity e is observed, according to our offline analysis the complexity is $O(mp_e)$ where p_e is the number of attributes for entity e (usually small in practice). Note that we must add new rows and columns to $\mathbf{L}$ and $\mathbf{X}$ as necessary, but this operation scales with the number of nonzero matrix elements when implemented with sparse matrices. Therefore, the total complexity is $O(m + np + mp_e) = O(mp_e + np)$.

4 INTRINSIC EVALUATION

Our first mode of evaluation is intrinsic: We obtain activity relatedness judgments from people *on their own data*, which is crucial because such judgments are nuanced and depend on knowing context and lived experience [10]. Though limited to a small set of individuals, our intrinsic evaluation allows us to directly characterize our approach. We complement it with a large-scale extrinsic evaluation in § 5, which measures downstream task performance on a public dataset in lieu of direct judgments on private data.

It is important to note that we are not aware of any prior work that evaluates graph representations via pairwise human judgments. We hope the present work invites further discussion from the community on intrinsic evaluations of graph representations.

4.1 Data

4.1.1 Data collection and preprocessing. We developed an on-device logging application that all task participants (§ 4.2.1) installed on their primary work computers at least two days prior to the rating task. The application indexes all emails and calendar appointments previously downloaded to the participant's machine, and further records the participant's interactions with these and other personal information items on her desktop. Metadata of these items include, e.g., the people associated with an email, the textual content of a file, when an individual clicked on a meeting, how long she focused on a web page, etc. Importantly, all logs are stored locally, the logging tool does not upload any information to the cloud, and all evaluation scripts using these logs were run locally on participants' computers from a USB drive. We collected only aggregate task performance metrics (§ 4.2.5) from each participant.

For preprocessing, we discard placeholder emails/appointments (e.g., "automatic reply"), emails/appointments from senders that the participant never personally contacted, emails without the participant on the *To*, *From*, or *CC* lines, emails that the participant only sent to herself, and, following [25], emails/appointments with over 10 recipients. To capture a rough notion of "importance", we retain only web documents/files that the participant dwelled on for at least 10 consecutive seconds. As textual attributes, we extract noun phrases from email/appointment subject lines and document/file titles. We remove general and domain-specific stopwords (e.g., filename extensions like "pdf", email abbreviations like "fwd") and phrases that often appear in search results ("Google Search").

4.1.2 Graph construction. Each personal web consists of *Email*, *Calendar Appointment*, *Web Document*, *File*, and *Contact* entities. We define the following edge relations: **(1)** *Contact-Email*, connecting people to emails that they sent, received, or were CC'ed on; **(2)** *Contact-Calendar Appointment*, connecting people to calendar appointments that they organized or attended; **(3-4)** *Email-Web Document* and *Calendar Appointment-Web Document*, connecting emails and appointments to web documents if the participant accessed the document immediately after reading the email or appointment (e.g. when clicking a link in the email body); **(5-6)** *Email-File* and *Calendar Appointment-File*, connecting emails and appointments to desktop files if the participant accessed the document immediately after reading the email or appointment; **(7)** *Email-Email*, connecting pairs of emails that appeared consecutively in a thread (i.e., replies). We construct each personal web from the participant's two most recent months of data for ease of contextualization.

4.2 Task definition

4.2.1 Participants. We recruited $n = 10$ participants (5 female, 5 male, ages 18-54) from a large enterprise technology company. Participants **P1**-**P4** were interns, participants **P5**-**P8** were software engineers, and participants **P9** and **P10** were senior researchers or research managers. Each participant was paid $25 upon completion of the task, which took about 30 minutes on average. The number of entities in each participant's personal web is given in Table 2.

4.2.2 Setup. Our task presents participants with pairs of their own personal information objects, and asks them to relate those pairs of objects in the context of their activities. Per system to be evaluated (§ 4.2.4), we identify pairs of related objects by following an information retrieval-style approach: Given a randomly sampled entity e (the "query"), the system constructs an ordered list of $k \leq 5$ candidate entities $e' \neq e$ it believes are "related" in terms of the participant's activities (to diversify results, we return only one email per email thread). Participants performed the task through a locally hosted web application. To fit the allotted time of the task, as each pair took about 30-45 seconds to rate, we limited the number of ranking systems and query entities such that each participant rated up to 60 pairs: 3 query entities $e \times$ up to 5 candidate related entities $e' \times 4$ ranking systems (§ 4.2.4).

4.2.3 Questionnaire. Each pair of entities displayed to the participant is associated with two questions:

- **Question 1** asks *how* the pair of items are related ("Why do you think the system related this pair of entities?"). The answer choices are: **(1)** *Low-level*, "These entities correspond to the same short-term task, appointment, or goal (e.g., a meeting, a TODO)"; **(2)** *Mid-level*, "These entities correspond to the same long-term project or activity (e.g., a research project, a home remodel)"; **(3)** *High-level*, "These entities correspond to the same general life category, not necessarily with defined start or end dates (e.g., *Personal*, *Professional*, *School*)"; **(4)** *Other*, "These entities are related for reasons not listed above"; **(5)** *Not related*, "The system is wrong. I cannot find any relationship between these entities"; and **(6)** *Unsure*, "The system may have its reasons, but I don't recognize one or more of these entities".
- **Question 2** asks the participant to assess the degree of activity-specific "relatedness" of the displayed pair ("In your opinion, how related is this pair of entities?"). The choices form a graded scale, scored as follows: *Strongly related* (4 points), *related* (3 points), *somewhat related* (2 points), *a little related* (1 points), and *not related at all* or *unsure* (0 points).

4.2.4 Systems compared. Due to the allotted time and cost of the task, and the fact that all scripts ran locally on personal machines, we restricted the task to two variants of our approach and two promising baselines. The **variants of our representations** rely on different activity-related attributes from text-bearing entities:

- **NP**: Seed entities' attributes are the noun phrases they contain, and their respective strengths are their frequencies.[2]
- **LSA [6]**: As attributes we take the rank-32 SVD of the document-phrase frequency matrix, treating each dimension of the decomposed matrix as a topic.

We set $\lambda = 10^2$ for both variants to emphasize the graph structure. After learning representations $\hat{\mathbf{X}}$, we return query entity e's k nearest neighbors e' in vector space, ranked in increasing order of Euclidean distance to the query entity e's vector representation. We compare to **two baselines** selected out of a number of approaches

[2]For this variant we do not differentiate between the offline and online versions of our representations, since they yield the same results.

Table 1: Average performance metrics per system averaged across all participants. Highest score among systems per metric shaded. Top group of rows: Averages across all pairs rated by participants as *a little related* or above. Bottom group of rows: Averages for pairs rated as *strongly related* only.

	Recall	Prec@1	Prec@2	Prec@3	Prec@4	Prec@5
***A little related, somewhat related, related,* and *strongly related* pairs**						
`People Overlap`	0.450 ± 0.11	0.933 ± 0.13	0.933 ± 0.11	0.922 ± 0.11	0.933 ± 0.10	0.933 ± 0.11
`node2vec`	0.440 ± 0.10	0.900 ± 0.21	0.867 ± 0.16	0.844 ± 0.17	0.858 ± 0.14	0.867 ± 0.14
`Ours-NP`	0.444 ± 0.07	0.967 ± 0.10	0.933 ± 0.11	0.911 ± 0.12	0.900 ± 0.11	0.867 ± 0.13
`Ours-LSA`	0.478 ± 0.07	1.000 ± 0.00	0.983 ± 0.05	0.944 ± 0.10	0.925 ± 0.10	0.907 ± 0.12
***Strongly related* pairs only**						
`People Overlap`	0.319 ± 0.11	0.370 ± 0.25	0.370 ± 0.20	0.290 ± 0.17	0.265 ± 0.14	0.247 ± 0.15
`node2vec`	0.447 ± 0.25	0.333 ± 0.31	0.352 ± 0.32	0.333 ± 0.27	0.306 ± 0.19	0.274 ± 0.16
`Ours-NP`	0.507 ± 0.24	0.519 ± 0.28	0.481 ± 0.21	0.420 ± 0.21	0.398 ± 0.21	0.356 ± 0.20
`Ours-LSA`	0.522 ± 0.23	0.407 ± 0.26	0.407 ± 0.19	0.383 ± 0.20	0.380 ± 0.21	0.356 ± 0.18

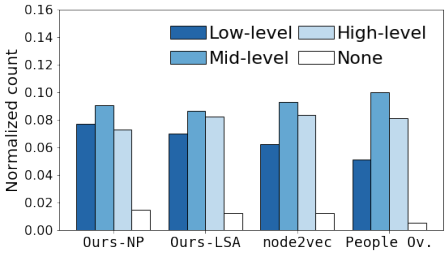


Figure 4: Question 1 answers by system across all participants.

Table 2: Average relatedness grade (Question 2) out of 4 across participants P1-P10. Parentheses: Each system's rank per participant; lower is better. Top: All entity pairs; Bottom: *Email-Email* pairs only. Last column: Grades and ranks averaged across all participants.

Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Avg. grade (rank)
# entities in G	157	258	320	303	256	291	203	232	1468	1637	
All pairs of entities											
`People Overlap`	2.00 (4)	**2.47 (1)**	2.67 (4)	1.87 (4)	**2.77 (1)**	**2.00 (1)**	2.00 (2)	2.00 (3)	2.43 (3)	2.13 (3)	2.22 ± 1.23 (2.60)
`node2vec`	**2.33 (1)**	2.40 (2)	3.07 (3)	1.93 (3)	2.33 (2)	1.87 (2)	1.80 (3)	1.93 (4)	2.20 (4)	1.73 (4)	2.16 ± 1.38 (2.80)
`Ours-NP`	2.27 (2)	1.93 (4)	**3.53 (1)**	**2.13 (1)**	2.27 (3)	1.87 (2)	1.80 (3)	**2.53 (1)**	**2.73 (1)**	2.60 (2)	2.37 ± 1.43 (**2.00**)
`Ours-LSA`	2.13 (3)	2.13 (3)	3.27 (2)	2.07 (2)	2.27 (3)	1.87 (2)	**2.27 (1)**	2.47 (2)	2.53 (2)	**2.80 (1)**	2.38 ± 1.38 (2.10)
***Email-Email* pairs only**											
`People Overlap`	2.60 (2)	**2.67 (1)**	2.44 (4)	1.75 (3)	2.55 (4)	1.69 (4)	**2.20 (1)**	2.33 (3)	2.46 (2)	2.13 (3)	2.26 ± 1.30 (2.70)
`node2vec`	2.60 (2)	1.88 (3)	2.78 (3)	1.80 (2)	**3.71 (1)**	**2.00 (1)**	1.00 (3)	1.62 (4)	2.14 (4)	1.73 (4)	2.07 ± 1.39 (2.70)
`Ours-NP`	2.40 (4)	1.83 (4)	**3.29 (1)**	1.67 (4)	3.62 (2)	**2.00 (1)**	1.00 (3)	**2.57 (1)**	**2.50 (1)**	2.33 (2)	2.40 ± 1.40 (2.30)
`Ours-LSA`	**2.80 (1)**	2.29 (2)	2.88 (2)	**2.00 (1)**	3.62 (2)	1.89 (3)	2.11 (2)	2.43 (2)	2.42 (3)	**2.79 (1)**	2.54 ± 1.30 (**1.90**)

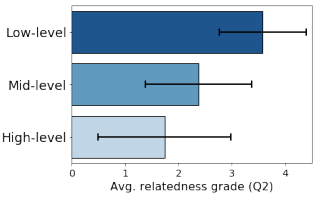


Figure 5: Question 2 answer averages stratified by answers to Question 1.

from retrieval, clustering, and embedding based on their performance in an independent pilot study involving six participants:

- **`People Overlap`** [7]: For each query entity e, we compute the Jaccard similarity between the people involved in e and all other entities e' excluding the participant, then rank the entities e' by similarity score and return the top k.
- **`node2vec`** [14]: We choose `node2vec`, which is based on the word2vec architecture [23], among other graph embeddings for its widespread usage and lightweight time and space complexity. For this baseline we first augment each participant's personal web by including nodes representing noun phrases and edges connecting entities to the noun phrases that they contain, similar to [1, 29]. We then apply `node2vec` on the augmented graph using its default parameter settings [14]. For each query entity e, we find its k nearest neighbor entities e' in embedding space.

4.2.5 Performance metrics. Per participant we collect only aggregate system performance metrics from the judgment task. Given a threshold of relatedness from the participant's answers to Question 2 (e.g., *a little related* up to *strongly related*), we compute recall and precision@k. When computing recall, we form the ideal set of related entities by pooling judgments across methods and baselines.

4.3 Results and discussion

Table 1 gives performance metrics averaged across all study participants. The first group of rows in Table 1 uses a permissive binarization of the "relatedness" scale (Question 2), considering pairs to be related unless the participant responded *unsure* or *not related at all*. The second group of rows uses a stricter binarization of relatedness, considering pairs to be related only if the participant chose *strongly related*. It is evident from Table 1 that all systems are able to identify related entities when the definition of "relatedness" is relatively loose. Our `LSA` representations mostly perform best here except for lower ranks of precision, where `People Overlap` performs best. However, with a stricter definition of relatedness, our `NP` representations mostly perform best.

It should be noted that the standard deviations in Table 1 are relatively high. We hypothesize that the variance can be partially explained by participants' varying roles in the workplace, which we found to be correlated with interpretations of the term "activity". For example, our representations performed well for the senior-level participants with many ongoing activities, whereas the `People Overlap` baseline worked well for junior-level employees with fewer ongoing professional activities. We found that senior-level participants (e.g., principal researchers, managers) tended to have many ongoing activities involving the same group of people, so more fine-grained textual cues were needed to distinguish these activities. By contrast, participants with fewer ongoing professional activities (e.g., interns, individual contributors) tended to view their activities as shaped primarily by the people involved.

We explore this effect further in Table 2, where we demonstrate each system's performance *per participant* along with the size of

each participant's personal web G as measured by the number of entities in the graph. In Table 2, the first group of rows gives average grades for Question 2 (out of 4) and average system ranks (lower is better) for all pairs of rated pairs. We find that our NP and LSA representations perform the best overall, whereas the performance of node2vec is middling and People Overlap is more polarized. The second group of rows gives the same information for *Email-Email* pairs only. Here our LSA representations perform by far the best, likely because taking the SVD of the document-term matrix better groups entities into high-level "topics" [6].

Figure 4 shows *what kinds of related* items each system found according to participants' responses to Question 1 ("Why do you think the system related this pair of items?"), with answers aggregated across all participants. All methods found a plurality of *mid-level related* pairs (e.g., participating in the same long-term project or activity), but NP found the most pairs with *low-level* (e.g., short term task) relationships. By contrast, People Overlap found the fewest *not related* pairs, which is intuitive as it is a high-precision baseline (Table 1, top). We also find that participants' responses to Question 1 correlate with their responses to Question 2 ("In your opinion, how related is this pair of items?"), as demonstrated by Figure 5. In particular, across all participants, there appeared to be a strong consensus that short-term tasks corresponded to the strongest relations between entities, with *low-level related* pairs of entities receiving 3.579 ± 0.82 points (out of 4) for Question 2, on average. This correlation suggests that while individuals may have interpreted the notion of "activity" in different ways, their ratings for Questions 1 and 2 were consistent.

Overall, our results demonstrate the strengths of our graph-based representations over several strong baselines. In particular, while there may not be a "one-size-fits-all" approach to activity discovery, we find that our representations perform well for people with many ongoing activities, e.g., senior-level employees. Future work could further investigate how professional roles correlate with individuals' perceptions of, and participation in, activities.

5 EXTRINSIC EVALUATION

To complement our small-scale intrinsic evaluation, we conduct additional experiments on a large public email dataset to demonstrate the versatility of our unsupervised representations.

5.1 Data

We use the Avocado email dataset[3], which comprises the inboxes of several hundred employees of a now-defunct IT company referred to as Avocado. We filter each inbox following the processing described in § 4.1. From each filtered inbox we extract a graph consisting of *Email*, *Contact*, and *File* entities. We define the *Contact-Email* and *Email-Email* relations the same way as in § 4.1.2, and define an *Email-File* relation that connects emails to their attachments. Table 3 provides aggregate statistics of all personal webs. Following our findings in § 4.3 on role-based differences of individuals' activities, we stratify our dataset by size, roughly along classes of professional roles (e.g., worker vs middle management vs officer) [19]: *Small* inboxes have 200–1 000 nodes in their respective graphs, *Medium* have 1 000–10 000, and *Large* have ≥10 000.

[3]https://catalog.ldc.upenn.edu/LDC2015T03

Table 3: Aggregates from the Avocado inboxes, stratified by size, after filtering and preprocessing following § 4.1. All personal web statistics are medians.

	Personal web		Entity types		
	# nodes	# edges	# emails	# contacts	# files
Small ($n = 45$)	501	1024	373	39	71
Medium ($n = 76$)	2862	6160	2 414	135	335
Large ($n = 7$)	12 759	37 119	10 755	303	1 457

5.2 Task definition

Following prior work in activity modeling [25], we use an email recipient recommendation task to evaluate the downstream utility of our graph-based representations.

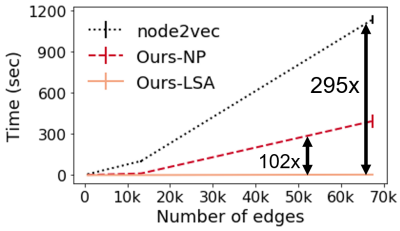
5.2.1 Setup. Per employee in the Avocado dataset, we construct a test set consisting of the last 8 months of emails from his or her inbox. We retain only the first email per email thread. For each test email, we remove the last recipient on the To line, following the setup in [25]. We discard emails with fewer than two recipients and emails whose last recipient was never seen in the training set. For the graph-based methods, we construct a graph given the missing edges between emails and contacts, and learn the respective model over the partial graph. At test time, given an email with the last recipient missing, each approach creates a ranked list of candidate recipients r', excluding the test email's sender s and observed recipients r. For the methods that represent entities as vectors, we return the candidate recipients r' in order of increasing Euclidean distance from the test email's vector representation. Following [25], our metrics of choice are hits@k for $k \in \{1, 2\}$, which quantifies the fraction of predictions where the correct recipient is ranked in the top k results, and mean reciprocal rank (MRR).

5.2.2 Methods compared. We learn our graph-based representations with the following variants:

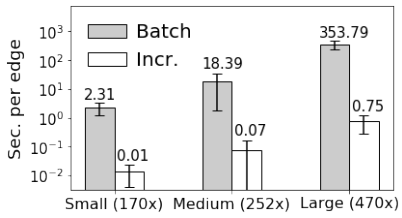
- **NP**: Using the same approach as § 4.2.4, we vary $\lambda \in \{10^{-1}, 10^{0}, 10^{2}\}$.
- **LSA**: Same as § 4.2.4, with $\lambda = 10^2$.
- **LDA**: We decompose the document-term matrix with Latent Dirichlet Allocation (LDA [3]) before propagation. We set $\lambda = 10^2$ and the number of latent topics to 10.

We compare to the following baselines:

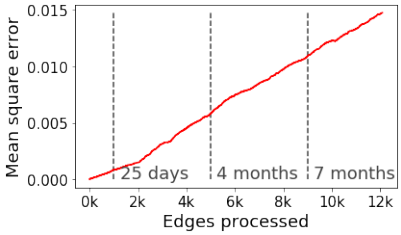
- **Random**: We rank the recipients r' in random order.
- **Frequent Recipients**: We rank recipients r' by $P(r' = u')$ for all people u' observed in the inbox.
- **Conditioned On Sender**: Similar to Frequent Recipients, but conditioned on sender s, e.g. $P(r' = u'|s)$.
- **Average NP**: We represent each email as a noun-phrase frequency vector and each candidate recipient u' as the average of all vectors representing emails she has sent or received.
- **node2vec**: We report the best node2vec performer, as described in § 4.2.4, on a grid search over the walk length $l \in \{10, 80\}$, in-out parameter $q \in \{1, 2\}$, and embedding dimension $d \in \{32, 128\}$. All other parameters are set to their defaults.



(a) Average time in seconds to train our NP and LSA representations offline versus offline node2vec as the graph size increases, measured by the number of edges.



(b) Log-scale training time for online and offline NP in seconds per edge, averaged across all inboxes of each size. Average efficiency gain in parentheses.



(c) Mean square error of a set of static NP representations learned once and not updated as new edges arrive over the course of a year for a *Medium* Avocado graph.

Figure 6: Offline and online learning of entity representations on the Avocado dataset.

Table 4: Performance in the recipient recommendation task averaged across all Avocado inboxes (Table 3). Top performer(s) per metric shaded. ▲: Significant over all methods not marked with † for a two-sided t-test at $p < 0.01$.

	Hits@1	Hits@2	MRR
Random	0.019 ± 0.023	0.038 ± 0.040	0.081 ± 0.060
Freq. Recipients	0.107 ± 0.106	0.184 ± 0.136	0.229 ± 0.105
Cond. On Sender	0.143 ± 0.094†	0.247 ± 0.113▲	0.282 ± 0.090†
Average NP	0.128 ± 0.088	0.209 ± 0.119	0.259 ± 0.102
node2vec	0.062 ± 0.072	0.092 ± 0.108	0.126 ± 0.114
Ours-NP, $\lambda = 10^{-1}$	0.111 ± 0.059	0.182 ± 0.096	0.225 ± 0.082
Ours-NP, $\lambda = 10^{0}$	0.158 ± 0.084▲	0.247 ± 0.105▲	0.290 ± 0.089▲
Ours-NP, $\lambda = 10^{2}$	0.143 ± 0.085†	0.225 ± 0.112†	0.267 ± 0.093†
Ours-LSA	0.110 ± 0.093	0.180 ± 0.126	0.224 ± 0.111
Ours-LDA	0.082 ± 0.080	0.141 ± 0.123	0.189 ± 0.111

5.3 Results and discussion

As shown by Table 4, our graph-based representations outperform or tie all baselines, suggesting their versatility in activity-centric tasks for which they are not directly optimized. Specifically, our NP representations with $\lambda = 10^0$ perform best on average across all Avocado inboxes, tied with the Conditioned On Sender baseline for Hits@1 but otherwise around 1 percentage point or higher than the best baseline. Our NP representations outperform our LSA representations by a significant margin in the recipient recommendation task. As demonstrated in § 4.3, our NP representations are best at identifying strongly-related pairs of entities (Table 1, bottom), which we hypothesize may be most useful for email recipient recommendation. Interestingly, we find that node2vec performs poorly, which suggests that it is ill-suited to small personal information collections. We hypothesize that its use of random walks may introduce spurious relations between entities in these graphs.

These results also offer insight into the effects of constructing graphs from personal information collections, which is a key design choice of this work. For example, a larger value of the regularization hyperparameter λ translates into more similar entity representations for entities that are closely connected in the graph. In the context of recipient recommendation, this corresponds to people who co-occur often on emails, leading to better prediction performance. Moreover, our Average NP baseline, which represents each person as an average noun phrase frequency vector, can be seen as a non-graph-based version of our NP representation. Our NP representations at $\lambda = 10^0$ outperform it significantly, suggesting that explicitly utilizing the underlying graph structure of the data may capture more information than pooling (e.g., averaging, concatenating) document vectors.

6 SCALABILITY EVALUATION

Finally, we examine how model training scales in both offline and online settings. All experiments were run on a single personal laptop with an Intel i7 1.90GHz processor and 16GB RAM.

6.1 Offline setting

We compare the efficiency of offline inference with our NP and LSA variants versus node2vec. We randomly sample one Avocado inbox of each size (Table 3) and train NP, LSA, and node2vec five times over each inbox, reporting the average training time in seconds in Figure 6a. We find that learning our NP representations is consistently faster than training node2vec (between 3× to 8×) since the latter requires computing expensive random walks across the graph in order to generate context for each node. Our LSA variant is even more efficient, up to 102× faster than NP, and up to 295× faster than node2vec, running in only a few seconds on a graph of 70k edges. This is because taking the rank-r SVD, which is $O(r^2 n)$ for n nodes in the graph [15], results in a constant number of (latent) attributes r being propagated across the graph. For a graph of m edges, the overall complexity of the LSA variant is $O(r^2 n + rm) = O(m)$ for a constant $r \ll n < m$.

6.2 Online setting

We next measure the difference between online and offline model inference. Recall from § 3.3.2 that our LSA variant is not eligible for incremental training because the SVD of the document-term matrix must be recomputed from scratch each time new data arrive, so we only experiment with the NP representations here. Figure 6b reports the average number of seconds per edge processed, and the average efficiency gain, on all Avocado corpora stratified by size (as detailed earlier in Table 3), for our offline- and online-learned NP representations. Online NP, which updates representations incrementally per new edge, is up to 470× faster than offline NP, taking on average less than 1 second per edge on the *Large* corpora.

Figure 6c demonstrates how the error of a single, *offline*-learned set of NP representations increases as new edges arrive over time for a randomly sampled *Medium* Avocado inbox. Here we measure the difference between the batch-learned offline representations and the most current, online-learned version with mean square error. Evidently the error increases approximately linearly as edges arrive, suggesting that for individuals with a high volume of incoming data (e.g., upper-level roles like managers, executives), updating the representations of their personal information objects in an online manner becomes more important.

7 DISCUSSION AND CONCLUSION

This paper proposes a graph-based approach to learning representations of items in the personal web, and presents an efficient online method for updating those representations as new data arrive. We demonstrate that our representations capture personal notions of activity-based relatedness, and support the downstream task of email recipient prediction. While our work is a step toward higher-level machine understanding of individuals' activities, many future research directions remain. For example, due to resource constraints, our human judgment task is limited by its number of participants and the number of pairs rated by each participant. Further means of human-centric evaluation include surveys to characterize people's perspectives toward activities, and longitudinal studies to capture performance via implicit feedback among a larger participant pool in an online setting (e.g., [31, 32]).

Another direction for future work is that of developing intrinsic evaluation tasks beyond the absolute graded ratings used in our study. For example, it has been shown that people find it easier to provide A/B preference judgments than absolute judgments [5]. In our context, such relative judgments would alleviate some issues with varying interpretations of the rating task. Future work could also develop subjective evaluation metrics tailored to the rating task, for example the "usefulness" or "surprisingness" (c.f. [36]) of a pair of related entities. Finally, we believe that our work demonstrates the promise of graph-based approaches toward activity discovery. We hope that future work in this direction will build upon the foundations we provide, ultimately toward artificial intelligence that helps people better structure and organize their lives.

ACKNOWLEDGEMENTS

This material is partially supported by the National Science Foundation under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, and an NSF Graduate Research Fellowship.

REFERENCES

[1] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from user interactions in personal search via attribute parameterization. In *WSDM*. ACM, 791–799.
[2] Jan R Benetka, John Krumm, and Paul N Bennett. 2019. Understanding Context for Tasks and Activities. In *CHIIR*. ACM, 133–142.
[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR* 3, Jan (2003), 993–1022.
[4] Vannevar Bush. 1945. As we may think. *Atlantic Monthly* 176 (1945).
[5] Ben Carterette, Paul N Bennett, David Maxwell Chickering, and Susan T Dumais. 2008. Here or there. In *ECIR*. Springer, 16–27.
[6] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JASIST* 41, 6 (1990), 391–407.
[7] Mark Dredze, Tessa Lau, and Nicholas Kushmerick. 2006. Automatically classifying emails into activities. In *IUI*. ACM, 70–77.
[8] Mark Dredze and Hanna Wallach. 2008. User models for email activity management. In *IUI Workshop on Ubiquitous User Modeling*.
[9] Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C Robbins. 2003. Stuff I've seen: a system for personal information retrieval and re-use. In *SIGIR*. ACM, 72–79.
[10] Susan Dumais, Edward Cutrell, Jonathan J Cadiz, Gavin Jancke, Raman Sarin, and Daniel C Robbins. 2016. Stuff I've seen: a system for personal information retrieval and re-use. In *SIGIR Forum*, Vol. 49. ACM, 28–35.
[11] Susan Dumais, Edward Cutrell, Raman Sarin, and Eric Horvitz. 2004. Implicit queries (IQ) for contextualized search. In *SIGIR*, Vol. 4. 594–594.
[12] Victor M González and Gloria Mark. 2004. Constant, constant, multi-tasking craziness: managing multiple working spheres. In *CHI*. ACM, 113–120.
[13] Catherine Grevet, David Choi, Debra Kumar, and Eric Gilbert. 2014. Overload is overloaded: email in the age of Gmail. In *CHI*. ACM, 793–802.
[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.
[15] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.
[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[17] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *KDD*. ACM, 538–543.
[18] Di Jin, Mark Heimann, Ryan A. Rossi, and Danai Koutra. 2019. Node2bits: Compact Time- and Attribute-aware Node Representations for User Stitching. In *ECML/PKDD*.
[19] Di Jin, Mark Heimann, Tara Safavi, Mengdi Wang, Wei Lee, Lindsay Snider, and Danai Koutra. 2019. Smart Roles: Inferring Professional Roles in Email Networks. In *KDD*. ACM, 2923–2933.
[20] William Jones. 2007. Personal information management. *Annual review of information science and technology* 41, 1 (2007), 453–504.
[21] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[22] Julian McAuley and Jure Leskovec. 2014. Discovering social circles in ego networks. *ACM TKDD* 8, 1 (2014), 4.
[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.
[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM, 701–710.
[25] Ashequl Qadir, Michael Gamon, Patrick Pantel, and Ahmed Hassan Awadallah. 2016. Activity modeling in email. In *NAACL-HLT*. 1452–1462.
[26] Jianqiang Shen, Lida Li, Thomas G Dietterich, and Jonathan L Herlocker. 2006. A hybrid learning system for recognizing user tasks from desktop activities and email messages. In *IUI*. ACM, 86–92.
[27] Jack Sherman and Winifred J Morrison. 1950. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics* 21, 1 (1950), 124–127.
[28] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB* 4, 11 (2011), 992–1003.
[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *WWW*. 1067–1077.
[30] Jaime Teevan, William Jones, and Benjamin B Bederson. 2006. Personal information management. *Commun. ACM* 49, 1 (2006), 40–43.
[31] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *SIGIR*. ACM, 115–124.
[32] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *WSDM*. ACM, 610–618.
[33] Steve Whittaker, Tara Matthews, Julian Cerruti, Hernan Badenes, and John Tang. 2011. Am I wasting my time organizing email?: a study of email refinding. In *CHI*. ACM, 3449–3458.
[34] Steve Whittaker and Candace Sidner. 1996. Email overload: exploring personal information management of email. In *CHI*. ACM, 276–283.
[35] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational context for ranking in personal search. In *WWW*. 1531–1540.
[36] Qian Zhao, Paul N Bennett, Adam Fourney, Anne Loomis Thompson, Shane Williams, Adam D Troy, and Susan T Dumais. 2018. Calendar-aware proactive email recommendation. In *SIGIR*. ACM, 655–664.
[37] Xiaojin Zhu. 2005. *Semi-supervised Learning with Graphs*. Ph.D. Dissertation.