Multi-Type Itemset Embedding for Learning Behavior Success

Daheng Wang
University of Notre Dame
Notre Dame, Indiana
dwang8@nd.edu

Meng Jiang
University of Notre Dame
Notre Dame, Indiana
mjiang2@nd.edu

Qingkai Zeng
University of Notre Dame
Notre Dame, Indiana
qzeng@nd.edu

Zachary Eberhart
University of Notre Dame
Notre Dame, Indiana
zeberhar@nd.edu

Nitesh V. Chawla
University of Notre Dame
Notre Dame, Indiana
nchawla@nd.edu

ABSTRACT

Contextual behavior modeling uses data from multiple contexts to discover patterns for predictive analysis. However, existing behavior prediction models often face difficulties when scaling for massive datasets. In this work, we formulate a behavior as a set of context items of different types (such as decision makers, operators, goals and resources), consider an observable itemset as a behavior success, and propose a novel scalable method, “multi-type itemset embedding”, to learn the context items’ representations preserving the success structures. Unlike most of existing embedding methods that learn pair-wise proximity from connection between a behavior and one of its items, our method learns item embeddings *collectively* from interaction among all multi-type items of a behavior, based on which we develop a novel framework, LearnSuc, for (1) predicting the success rate of any set of items and (2) finding complementary items which maximize the probability of success when incorporated into an itemset. Extensive experiments demonstrate both effectiveness and efficency of the proposed framework.

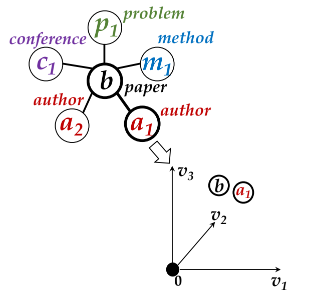
KEYWORDS

Behavior modeling, Representation learning, Behavior data embedding, Itemset embedding, Recommender systems

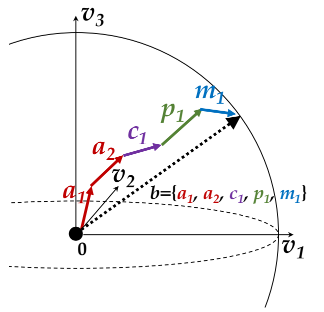
ACM Reference Format:
Daheng Wang, Meng Jiang, Qingkai Zeng, Zachary Eberhart, and Nitesh V. Chawla. 2018. Multi-Type Itemset Embedding for Learning Behavior Success. In *KDD ’18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219949

1 INTRODUCTION

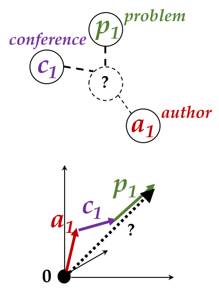
Behaviors are the products of the interaction between multiple types of contexts [25]. The *multi-type contexts* often include operators, goals, resources, spatiotemporal and social dimensions. Consider paper-publishing behavior as an example: it has authors, target conference/journal, datasets, problems, methods, references, and so on. Contextual behavior modeling has been broadly used for pattern discovery and predictive analysis. Tensor methods decompose multidimensional counts (e.g., numbers of co-occurrence among one author, one conference, and one reference) into low-dimensional latent vectors [20, 24], however, real behaviors do not guarantee one *context item* per dimension/type [22] – a paper can



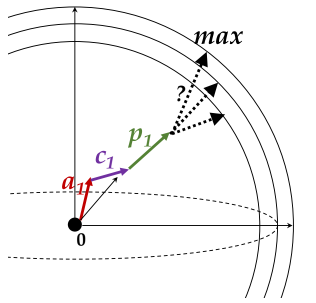
(a) Existing embeddings preserving pair-wise proximity of a behavior and an item



(b) Multi-type itemset embedding preserving set structures of behavior success



(c) Task I: Predict the probability of an itemset being observed in the future



(d) Task II: Recommend complementary items to maximize the probability

Figure 1: The proposed *multi-type itemset embedding* learns context items’ representations collectively from the itemset structure. When it is applied to paper-publishing behavior data, the embeddings preserve the composition of a paper (authors, datasets, problems, methods) being successfully published instead of pair-wise similarity between a paper and any item of it. This novel invention enables functionalities of two important behavior modeling tasks, contextual behavior prediction and behavior context recommendation.

have multiple authors and multiple references. Jamali et al. proposed a multicontextual factor model to learn latent factors of users and items for product recommendation [19], but neither tensor methods nor factor models can scale for massive behavior data.

Recently embedding methods have been proposed to efficiently learn low-dimensional vectors of nodes from large heterogeneous networks. Certainly we can explicitly represent behaviors and their context items as nodes and obtain a large "behavior-to-item" bipartite network, as shown in Figure 1(a). Network embeddings preserve *pair-wise proximities* between a behavior and any of its item, such as connections, common neighbors, and random walk-based measures. This methodology facilitates *similarity search* and *node categorization* with the pair-wise proximity-preserved vectors.

In this paper, we focus on an important property of behavior "behavior success": We represent a behavior as a multi-type itemset, i.e., a set of context items of multiple types, instead of a multidimensional count or an explicit node. For example, as shown in Figure 1(b), the paper-publishing behavior (actually a paper) is represented as an itemset $b = \{a_1, a_2, c_1, p_1, m_1\}$ where a_1 and a_2 are author names, c_1 is a conference name, p_1 is a research problem name, and m_1 is the proposed method name. The *success rate* of this behavior $\hat{r}(\cdot)$ is defined as the probability of it being occurred/observed, e.g., $\hat{r}(b) = 1$ (100%) if the paper is published; otherwise, $\hat{r}(b) \in [0, 1)$. This definition can be generalized as *the benefit a behavior brings*, for example, if a tweeting behavior is represented as an itemset of user, hashtags, words/phrases and emoticons, its success rate can be evaluated as its popularity, i.e., the number of times being retweeted.

Here come the novel tasks we would like to solve:

Task 1 (Contextual behavior prediction). **Given** a behavior b and its set of context items, **predict** b's success rate $\hat{r}(b)$.

For example, given a set of authors, keywords, references and a conference, predict the probability of a paper of these items being published in the conference's proceedings (see Figure 1(c)).

Task 2 (Behavioral context recommendation). **Given** a behavior b and its set of context items, **recommend** complementary context items that will maximize the success rate of this behavior.

For example, given a student author, a top-tier target conference, some keywords (e.g., a dataset, a problem), recommend other authors (colleagues or external collaborators), other proper keywords and references that will maximize the chance of the student's paper being accepted by the conference (see Figure 1(d)).

Note that these two tasks can be extended for many applications such as target advertising, personalized consulting, and military mission planning. Generally, we look for complementary operators and resources to maximize the probability of achieving the goal, incorporated with existing operators, resources, and other contexts.

Unfortunately, network embedding models cannot effectively solve these problems. We point out their weak points and propose our ideas as follows.

- Item representations should be learnt collectively from the co-occurrence of all multi-type context items in the behavior/itemset (or "itemset structure", or behavior success property) instead of the co-occurrence of item pairs.
- Item representations should preserve the success rate of behavior/itemset instead of pair-wise proximity between a behavior/itemset and an item in a network.

Therefore, we propose a novel *Multi-Type Itemset Embedding* method that efficiently learns representations of behavior's context items of multiple types preserving itemset structures. This is rather challenging. First of all, similarity between feature vectors of any pair of items in the itemset is no longer capable of measuring the success rate of the itemset. Intuitively, a behavior is more likely to be successful if the operators have complementary knowledge/skills rather than very similar knowledge/skills. Second, the itemset structure has heterogeneity: different context types contribute to the success at different extents. For example, it is absolutely more impactful for a paper to involve one more internationally known expert than one more relevant reference. Third, in most of the cases, we can only observe positive behaviors. For example, it is easy to collect tons of published papers but hard to find rejected or non-finished ones. The embedding method has to be careful to generate negative instances for representation learning.

To address the above challenges, our method measures the success rate of an itemset in the vector space by the hyperbolic tangent of the sum of vectors of items in the itemset. It learns items' feature vectors collectively on both *similarity (angle)* between any pair of vectors and *norm* of each vector itself by optimizing the co-occurrence of all the items in the set. Second, our method assigns type weights to the context items when representing the itemset as a weighted sum of vectors, which has been demonstrated to be useful in experiments. Third, we introduce two kinds of *negative behavior sampling* strategies, size-constrained and type distribution-constrained, to generate negative samples when they are unavailable. We propose a novel framework, called LearnSuc, that first learns items' low-dimensional representations using *Multi-Type Itemset Embedding*, and then feed the itemset representations into a logistic regression model to predict success rate or recommend items for maximizing the success rate.

We summarize our main contributions in this paper as follows.

- We propose to study two novel tasks in contextual behavior modeling, contextual behavior prediction and behavioral context recommendation, when it is straightforward to consider behavior records as multi-type context itemsets.
- We propose Multi-Type Itemset Embedding method to collectively learn items' low-dimensional representations preserving itemset structures. It includes a novel measurement of success rate for learning the itemset structures, considers type weights for heterogeneity, and conducts negative behavior sampling for representation learning.
- We develop the LearnSuc framework and our extensive experimental studies demonstrate its effectiveness and efficiency on both behavior modeling tasks.

The rest of this paper is organized as follows. Section 2 reviews related work and Section 3 defines concepts and research problems. Our LearnSuc framework is presented in Section 4. Section 5 shows experimental results. In Section 6, we conclude the paper and provide discussions on future work.

2 RELATED WORK

In this section, we review existing methods in three relevant fields to our work, including contextual behavior modeling, network data representation learning, and text data representation learning.

Contextual behavior modeling. There has been a wide line of research on learning latent representations of context items in behavior data towards various applications [3, 21, 34]. Agarwal et al. proposed localized factor models combining multi-context information to improve predictive accuracy in recommender systems [2]. Jamali et al. proposed context-dependent factor models to learn latent factors of users and items for recommendation [19]. Besides factor models, tensor decompositions have been widely used for modeling multi-contextual data [33]. Jiang et al. proposed a tensor-sequence decomposition approach for discovering multi-faceted behavioral patterns [20]. Ermiş et al. studied various alternative tensor models for link prediction in heterogeneous data [12]. Lian et al. proposed regularized tensor factorization for spatio-temporal recommendation [24]. Yang et al. developed a predictive task guided tensor decomposition model for representation learning from Electronic Health Records [41]. Perros et al. designed a scalable PARAFAC2 tensor model for large and sparse datasets [31]. However, the computational cost of factorizing a large-scale matrix or tensor is usually very expensive, and none of the existing behavior modeling methods can *efficiently* learn item representations to optimize *success rate* on massive behavior data.

Network data representation learning. Network embedding methods learn node representations that preserve node proximities (e.g., one-hop or two-hop connections) in network data [4, 9, 10, 40]. DeepWalk [30] used random walks to expand the neighborhood of a node and expected nodes with higher proximity yield similar representations. node2vec [14] presented biased random walkers to diversify the neighborhood. LINE [37] provided clear objectives for homogeneous network embedding that articulates what network properties are preserved. We have spotted a series of heterogeneous network embedding work [6, 11, 15, 18, 39] that capture heterogeneous structural properties in network data. If we explicitly represent behavior entries as nodes and thus behavior datasets are represented as behavior-item heterogeneous bipartite networks, existing network embedding methods can be applied to learn representations of both items and behaviors. However, these methods preserve node proximities so they can only find *similar* items for a behavior. Our proposed method learns the combinations of context items of each behavior and preserves success rate of the behavior. Therefore, it finds *complementary* items that will maximize the success of a target behavior. There are other network embedding methods designed to learn task-specific node representations by utilizing label information in supervised fashion such as PTE [36]. Chen et al. [7] proposed a task-guided and path-augmented heterogeneous network embedding model to identify author names given anonymized paper's title. The most recent work PNE [8] decomposes a partially labeled network into two bipartite networks and encodes the node label information by learning label and context vectors from the label-context network. In our case, we only have the success rate for behaviors, which can be binary or real values; and, we aims at learning the item representations preserving behavior success. Individual item label about success carries little meaningful information outside the context of behavior. Thus, these supervised methods cannot be directly applied.

Symbol	Description
c, $\mathcal{C}$	Context item, and the set of all items
t, $\mathcal{T}$	Context (item's) type, and the set of all types
b, $\mathcal{B}$	Behavior, and the set of all "positive" behaviors
$r(b)$, $\hat{r}(b)$	Estimated and observed success rate of behavior b
$t(c)$	Context type of context item c
d	Number of dimensions in a low-dimensional space
$\vec{c}$	Low-dimensional vector of context item c
$\vec{b}$	Low-dimensional vector of behavior b
w_t	Learning weight of context type t

Table 1: Symbols we use in this paper and their descriptions.

Text data representation learning. With the success of deep learning techniques, representation learning becomes popular starting from practices on text data. Mikolov et al. proposed the word2vec framework to learn the distributed representations of words in natural language [13, 26, 27]. Pennington et al. proposed GloVe to learn word vectors from nonzero elements in a word-word co-occurrence matrix [29]. Le et al. extended the embedded objects from words or phrases to paragraphs [23]. Recently, Nichel et al. proposed Poincaré embedding [28] based on a non-Euclidean space to preserve hierarchical semantic structures [5]. Our work focuses on representation learning from behavior data. Here behavior is represented as multi-type itemset instead of word sequences.

3 PROBLEM DEFINITION

In this section, we first define several concepts used throughout this paper and then formally define the research problems we study. Table 1 presents symbols we use in this paper and their descriptions.

Definition 3.1 (Context type). A behavior includes multiple types of contexts such as individuals who make the action, behavioral goals and resources. A **context type** is the type of a context. For example, a paper-publishing behavior has authors (as operators), a conference or journal (as behavioral goal), keywords, technical terms, datasets, and references (as resources). A tweet-posting behavior has a social network user (as operators), a specific topic (as behavioral goal), words, hashtags, urls, and emoticons (as resources). Any type of these elements can be a context type such as "author", "conference", "reference", "user", and "hashtag".

Definition 3.2 (Context item). A **context item** is a concrete item of a context, such as an author's name, a conference's name or a concrete publication as a reference in paper-publishing behaviors. A context item c must have a context type $t(c)$.

Definition 3.3 (Behavior and multi-type itemset). A **behavior** is a relationship among multiple types of context items. It can be represented as a set of the context items. Therefore, a behavior is equivalent to a **multi-type itemset**.

For example, given a conference paper, the paper-publishing behavior is a set of author(s), conference, keyword(s), reference(s) and some other relevant items to this paper. The quantity and types of items in an itemset may vary from behavior to behavior. A paper-publishing behavior contains at least one author and keyword, exactly one conference, and multiple references. A tweet-posting behavior contains exactly one user and at least one character.

Definition 3.4 (Success rate). The **success rate** of a behavior is the value (as a numeric label) denoting the real-world *success* of that behavior given its particular set of context items. For each set of context items which constitutes a complete behavior b, we use $\hat{r}(b)$ indicates its success, where $\hat{r}(b) \geq 0$ for any b.

For a specific kind of behavior, in order to define the success rate, we should first define what is **success**. For example, for each paper in a publication dataset, we say the set of context items in this paper makes a success of a paper-publishing behavior, and thus the success rate of this behavior is a positive number – we use 1 as default and we call the behavior a **"positive" behavior**. This presents a challenge, as most datasets do not include entries on unpublished works. We will overcome this obstacle in the model by adopting the strategy of negative sampling. Basically, we assume that most of the *non-existing multi-type itemsets* indicate unsuccessful behaviors. We denote them as **"negative" behaviors** and set their success rate as 0.

Given different measurement of success, the success rate could be different. For tweet-posting behaviors, if the success is measured as the existence of their context itemsets in tweet data, we have tons of positive behaviors but no real negative ones; we have to generate non-existing itemsets as negative behaviors. If the success is measured as the behavior's popularity, the success rate is the number of views, likes, retweets, or shares [17]. In this case, positive behaviors are popular posts and negative behaviors are unpopular but still real posts. No non-existing itemset needs to be generated though the success rate may need to be normalized.

In summary, the success rate associated with each behavior may be explicit (e.g., a rating or score that the behavior received), or it may be implicit in the behavior data (e.g. the number of occurrences of the behavior). Now we define what is behavior data.

Definition 3.5 (Behavior data). **Behavior data** is defined as $D = (C, \mathcal{T}, \mathcal{B})$, where C is the set of unique context items, each having a particular type in $\mathcal{T}$, and $\mathcal{B}$ is the set of behaviors, each representing a relationship among one or more context items. Each behavior $b \in \mathcal{B}$ is a set of context items $b \subset C$, which is associated with a nonnegative, normalized, observed success rate $\hat{r}(b)$.

If both $|C|$ and $|\mathcal{B}|$ are big, we call the dataset *massive behavior data.* Based on the above concepts, we formally define the research problem as below.

Problem 1 (Massive behavior data embedding)**.** Given a massive behavior dataset $D = (C, \mathcal{T}, \mathcal{B})$, the problem of **Massive Behavior Data Embedding** aims to represent each context item $c \in C$ as a low-dimensional vector $\vec{c} \in \mathbb{R}^d$, i.e., learning a function $f_D : C \rightarrow \mathbb{R}^d$, where $d \ll |C|$. In the space $\mathbb{R}^d$, the contributions of each context item towards behavior's success are preserved.

As in Definition 3.3, a behavior can be generalized to a multi-type itemset of the same structure. Therefore, the problem of behavior data embedding is equivalent to multi-type itemset embedding.

Problem 2 (Multi-type itemset embedding)**.** Given a large set of items C, their types $\mathcal{T}$ and a large set of multi-type itemsets $\mathcal{B}$, the problem of **Multi-type Itemset Embedding** aims to represent each item $c \in C$ as a low-dimensional vector $\vec{c} \in \mathbb{R}^d$, i.e., learning a function $f_D : C \rightarrow \mathbb{R}^d$, where $d \ll |C|$. In the space $\mathbb{R}^d$, each item's contribution towards itemset's composition is preserved.

Much like heterogeneous information network embedding, itemset embedding aims to represent context items of various types as vectors in a low-dimensional space. Unlike network embedding, which preserves pair-wise proximity between nodes, itemset embedding preserves the itemset structures. The vector representations of items within an itemset may not be close to each other in $\mathbb{R}^d$, but the vectors should sum to a vector with a magnitude representative of their behavior's success. In this way, the contributions of each context item towards behavior's success are preserved.

Unlike network embedding models such as DeepWalk [30], LINE [37] and node2vec [14] that preserve *proximities* and were evaluated on clustering tasks, our behavior data embedding model, also a multi-type itemset embedding model, preserves *success property.* We will evaluate it on the two tasks of behavior modeling we have introduced in Section 1 and compete with existing works in experiments. These two tasks are challenging and important in this era of witnessing how artificial intelligence and data science may significantly change human beings' decision-making process.

4 THE LEARNSUC FRAMEWORK

Our LearnSuc framework has two modules for learning behavior success. The first module is a multi-type itemset embedding model that learns item representations from behavior data, in which we propose a novel metric of measuring the success rate of a behavior. We will evaluate the complexity of this model. The second module is to feed the itemset representations into a logistic regression model to predict the probability of a future behavior's success. It can also recommend complementary items to maximize the probability.

4.1 Multi-Type Itemset Embedding

A desirable embedding model for behavior data must satisfy several requirements: first, it must be able to preserve the success property of multi-type itemsets; second, it must scale for massive behavior data, say millions of context items and behaviors; third, it must deal with behaviors with arbitrary types and quantity of context items. In this section, we present a novel multi-type itemset embedding model which satisfies all the three requirements.

4.1.1 Model Description. We describe the embedding model to preserve success properties of behaviors. The success of a behavior refers to the success achieved given a particular set of context items. For each behavior b as a particular set of multi-type context items, we define b's low-dimensional vector representation as follows:

$$\vec{b} = \sum_{c \in b} w_{t(c)} \cdot \vec{c} \in \mathbb{R}^d, \quad (1)$$

where $\vec{c} \in \mathbb{R}^d$ is the low-dimensional vector representation of context item c; $w_{t(c)}$ is the type weight of c's context type. Different item types may have different levels of contributions to the behavior's success. For example, one author or one keyword often contributes more to a paper's acceptance than one reference. Therefore, we assign type weights to the context types when we sum the items' low-dimensional vectors to represent the behavior. We will discuss type-weight parameter settings in the experiment section.

For a behavior b, we define b's estimated success rate as follows:

$$r(b) = \tanh \frac{\|\vec{b}\|_2}{2} = 2 \cdot \frac{1}{1 + e^{-\|\vec{b}\|_2}} - 1, \tag{2}$$

where $\|\vec{b}\|_2 \in [0, \infty)$ is the Euclidean norm of $\vec{b}$ in the d-dimensional space. The hyperbolic tangent function $\tanh(x)$ is a rescaled version of the *logistic sigmoid* function $g(x) = \frac{1}{1+e^{-x}}$: $\tanh(x) = 2g(2x) - 1$, and its output range is $[0, 1]$ instead of $[\frac{1}{2}, 1]$ because the norm $\|\vec{b}\|_2$ is nonnegative. Eqn. (2) defines a distribution $r(\cdot)$ over the behavior space $\mathcal{B}$, and its empirical success rate can be observed as $\hat{r}(b)$. To preserve the success properties, a straightforward way is to minimize the following objective function:

$$O = d(\hat{r}(\cdot), r(\cdot)), \tag{3}$$

where $d(\cdot, \cdot)$ is the distance between two distributions. We choose to minimize the KL-divergence of two success-rate probability distributions following the LINE model [37]. Replacing $d(\cdot, \cdot)$ with KL-divergence, we have:

$$O = -\sum_{b \in \mathcal{B}} \hat{r}(b) \log r(b). \tag{4}$$

Remark. Traditional embedding models optimize the similarity between vectors of a pair of items, i.e., $\vec{c_i} \cdot \vec{c_j}$. Our itemset embedding model optimizes the success rate based on the norm of an itemset's vector. If we expand the norm as below,

$$\|\vec{b}\|_2 = \left(\sum_{c \in b} {w_{t(c)}}^2 \cdot {\|\vec{c}\|_2}^2 + \sum_{\substack{c_i, c_j \in b \\ c_i \neq c_j}} w_{t(c_i)} w_{t(c_j)} \cdot \vec{c_i} \cdot \vec{cj} \right)^{\frac{1}{2}}, \tag{5}$$

it is worthwhile of showing that our method learns the item vectors collectively and it optimizes not only the vector similarity of every pair of items in the set but also the norm of each item's vector $\|\vec{c}\|_2$.

4.1.2 Model Optimization. As explained after Def. 3.4 in Sec. 3, models should be trained on the behavior dataset $\mathcal{B}$ differently according to different measurement of success. For example, in order to optimize the objective function in Eqn. (4), if the success of a tweet-posting behavior is measured as the behavior's popularity (e.g., the number of retweets), $\mathcal{B}$ has both positive (popular) and negative (unpopular) behaviors. If a behavior's success is a binary measurement of its existence given its complete set of context items, $\mathcal{B}$ has only positive behaviors, and we have to generate non-existing itemsets as negative behaviors. We will introduce our model optimization approaches respectively as follows.

When $\mathcal{B}$ has both positive and negative behaviors. Suppose the distribution of observed success rate $\hat{r}(.)$ (e.g., the number of times a tweet is retweeted) follows the Power Law, which is often applicable in the real world. We have a reasonable number of negative behaviors in $\mathcal{B}$, and the objective function for each behavior b

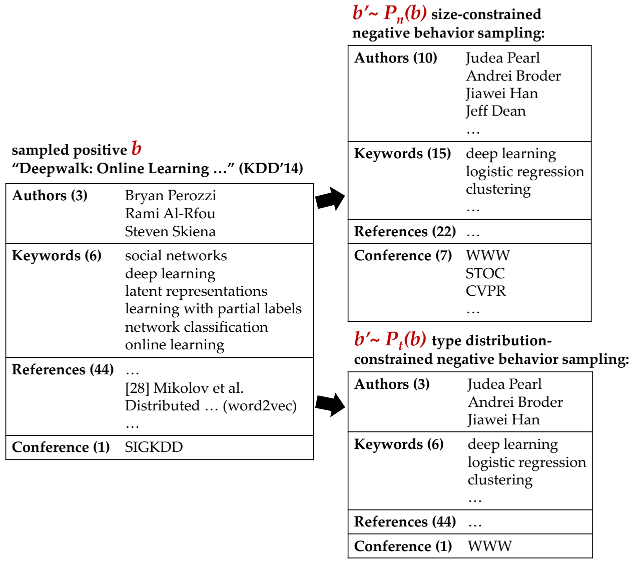


Figure 2: Negative behavior sampling: Given a sampled "positive" behavior (e.g., observable paper), we have two strategies to generate negative samples: one is size-constrained and the other is type distribution-constrained.

can be specified as follows:

$$\hat{r}(b) \log \tanh \frac{\|\vec{b}\|_2}{2} = \hat{r}(b) \log \tanh \frac{\| \sum_{c \in b} w_{t(c)} \cdot \vec{c} \|_2}{2}. \tag{6}$$

We adopt the asynchronous stochastic gradient algorithm (ASGD) [32] for optimizing Eqn. (6). In each step, the ASGD algorithm samples one behavior (either positive or negative) and then updates the model parameters. If a behavior b is sampled, the gradient w.r.t. the embedding vector $\vec{c}$ of a context item c in b will be calculated as:

$$\begin{aligned} \frac{\partial O}{\partial \vec{c}} &= \frac{\hat{r}(b)}{\sinh \|\vec{b}\|_2} \cdot \frac{\partial \|\vec{b}\|_2}{\partial \vec{c}} = \frac{\hat{r}(b)}{\sinh \|\vec{b}\|_2} \cdot \frac{\partial \|\vec{b}\|_2}{\partial \vec{b}} \cdot \frac{\partial \vec{b}}{\partial \vec{c}} \\ &= \frac{w_{t(c)} \hat{r}(b)}{\|\vec{b}\|_2 \sinh \|\vec{b}\|_2} \cdot \vec{b}. \end{aligned} \tag{7}$$

When $\mathcal{B}$ has only positive behaviors. In this case, we sample *non-existing multi-type itemsets* and use them as "negative behaviors". We approximate the optimization by sampling a mini-batch of behaviors which includes one positive sample and several negative samples. Inspired by the negative sampling technique proposed in [27], we propose the technique of negative behavior sampling on the behavior data (or multi-type itemset data) as shown in Figure 2.

Negative behavior sampling. Given the set of context items for each context type, how do we generate a non-existing multi-type itemset b' when we sample a positive itemset b from $\mathcal{B}$? We tried two different sampling strategies. The first strategy is to apply a simple size constraint on the entire itemset (say, the number of b''s items is the number of b's items) without constraining the number of items for any type. We first generate $n(t)$ that is the number of items of type $t \in \mathcal{T}$ so that the sum of $n(\cdot)$ is $|b|$, and then for each type t, we randomly select $n(t)$ context items from C_t which is the set of items of type t in C, and put into b'. We denote this *size-constrained negative behavior sampling* as $b' \sim P_n(b)$ and have the following objective:

$$\hat{r}(b) \log \tanh \frac{\|\vec{b}\|_2}{2} + \sum_{k=1}^{K} \mathbb{E}_{b' \sim P_n(b)} \log \tanh \frac{\|\vec{b'}\|_2^{-1}}{2}, \tag{8}$$

whose gradient is derived as follows:

$$\frac{\partial O}{\partial \vec{c}} = w_{t(c)} \cdot \left(I_{c \in b} \cdot \frac{\hat{r}(b)}{\|\vec{b}\|_2 \sinh \|\vec{b}\|_2} \cdot \vec{b} - I_{c \in b'} \cdot \frac{1}{\|\vec{b'}\|_2^{3} \sinh \frac{1}{\|\vec{b'}\|_2}} \cdot \vec{b'} \right), \tag{9}$$

where I_x is an indicator function that returns 1 if x is true and 0 if x is false. K is the number of negative behavior samples. Note that here $\hat{r}(b)$ is above zero. Specifically, we set $\hat{r}(b) = 1$ for paper-publishing behaviors.

The second strategy considers the context type distribution in a sampled positive itemset b. For each type t, we randomly select $n(t)$ context items from C_t, where $n(t)$ is the number of context items of the type t in the behavior b. We denote the *type-distribution-constrained negative behavior sampling* as $b' \sim P_t(b)$ and just replace P_n in the objective of Eqn. 9 and derivative of Eqn. 9 with P_t. We denote our method that uses size-constrained sampling as **LearnSuc-$\mathbf{P_n}$**, denote the method that uses type-distribution-constrained sampling as **LearnSuc-$\mathbf{P_t}$**, and compete with baseline methods in the experiments.

4.1.3 Complexity Analysis. Sampling an itemset from behavior data takes constant time, $O(1)$, and optimization with negative sampling takes $O(d(K+1))$ time, where d is the number of dimensions and K is the number of negative samples. So overall each step takes $O(dK)$ time. The number of steps is proportional to the number of behaviors/itemsets $O(|\mathcal{B}|)$. Therefore, the overall time complexity of the itemset embedding model is $O(dK|\mathcal{B}|)$, which is linear to the number of context items $|C|$.

4.2 Prediction and Recommendation Models

Once the representations of context items have been learnt by the multi-type itemset embedding model preserving the behavior success property, we can use those low-dimensional feature vectors to solve the two behavior modeling tasks we have given in Section 1.

4.2.1 Contextual Behavior Prediction. Given a behavior/itemset b and its set of context items, we train a logistic regression model with the itemset's representation $\vec{b}$ and label $\hat{r}(b)$ and apply the model to predict the success probability of testing instances.

4.2.2 Behavioral Context Recommendation. The above two models can both be applied for recommending complementary items for a potential behavior/itemset. The goal is to maximize the *predicted probability* of being successful/observed. It is time-consuming (exponential time) to enumerate all itemset candidates and compute their probability scores, so in the experiments we will hide only one item from each testing itemset. We will leave the task of recommending the best combination of multiple items as future work.

5 EXPERIMENTS

In this section, we first introduce the data we use, and then for each behavior modeling task, we present validation settings, quantitative analysis, and qualitative analysis, and finally we evaluate efficiency.

Table 2: The multi-type itemset embedding LearnSuc outperforms baselines on contextual behavor prediction. (The types are {author, conference, keyword, reference} in w_t.)

Method	Weights w_t	MAE	RMSE	AUC	F1
PCA [16]		.0897	.2228	.9814	.9432
LINE [37]		.0949	.2123	.9823	.9439
DeepWalk [30]		.0723	.1784	.9921	.9627
node2vec [14]		.0603	.1701	.9929	.9663
metapath2vec [11]		.0567	.1648	.9934	.9692
LearnSuc-P_n (Size-constrained negative behavior sampling)	{1,1,1,1}	.0492	.1439	.9956	.9793
	{1,1,1,3}	.0496	.1449	.9952	.9765
	{1,1,3,1}	.0503	.1489	.9952	.9734
	{1,3,1,1}	.0509	.1485	.9956	.9747
	{3,1,1,1}	.0504	.1475	.9954	.9742
LearnSuc-P_t (Type-distribution -constrained negative behavior sampling)	{1,1,1,1}	.0448	.1272	.9973	.9826
	{1,1,1,3}	.0506	.1384	.9959	.9796
	{1,1,3,1}	.0568	.1387	.9970	.9812
	{1,3,1,1}	.0462	.1313	.9965	.9825
	{3,1,1,1}	**.0432**	**.1243**	**.9976**	**.9847**

5.1 Dataset Description

We use a public dataset from the Microsoft Academic project to study the behavior of publishing a paper at a conference in the field of computer science. We collect 10,880 papers whose context items include one conference in the field of data science, at least one author, at least one keyword and at least one reference. Overall we have 317 conferences, 12,330 authors, 3,548 keywords, and 18,971 references. In average, each paper has 1 conference, 2.64 authors, 9.65 keywords, and 7.68 references (20.97 context items in total).

5.2 Baselines and Parameter Settings

Since there is not any *itemset embedding* method in the literature, we compare our LearnSuc with the dimensionality reduction methods [38] and the state-of-the-art *network embedding* models:

(1) PCA [16]: It learns data representations that describe as much of the variance in the data as possible.
(2) LINE [37]: This homogeneous network embedding method preserves the 1st- and 2nd-order of node proximity.
(3) DeepWalk [30]: It uses local information obtained from truncated random walks to learn latent representations of vertices in a network.
(4) node2vec [14]: It learns continuous feature representations for nodes in networks by maximizing the likelihood of preserving network neighborhoods of nodes.
(5) metapath2vec [11]: This method learns node representations based on meta-path-based random walks from heterogeneous networks.

We evaluate LearnSuc's alternatives such as LearnSuc-P_n and LearnSuc-P_t to compare the effectiveness of different negative sampling strategies. For all methods, we use the same parameters: The vector dimension d is 128; the size of negative samples is 10. The weights of context types are {3 (author), 1 (conference), 1 (keyword), 1 (reference)} as default. We also vary each of them for examining different settings of the parameters.

5.3 Contextual Behavior Prediction

5.3.1 Validation settings. We use 10-times hold out validation to evaluate the performance: We randomly sample 10% of the behaviors (papers) for testing and 90% for training. We use the whole training set to learn itemset embeddings and use 10% of the training set (as positive instances) along with the same amount of itemsets that are generated by type-distribution-constrained negative behavior sampling (as negative instances). For each method, we train a logistic regression model with the latent representations and label of the instances. We generate negative instances of the same size of the testing set and for each test instance b, the logistic regression model returns a probability score $r(b)$.

We use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to evaluate the predictions:

$$MAE = \frac{\sum_{b \in \mathcal{T}} |r(b) - \hat{r}(b)|}{|\mathcal{T}|}, \quad (10)$$

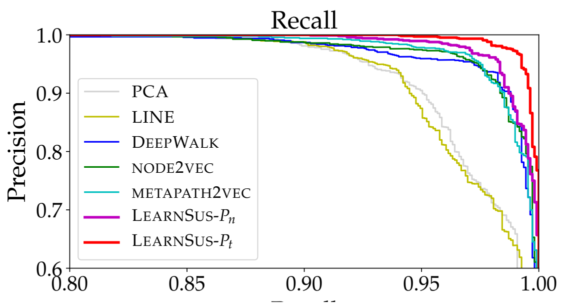
$$RMSE = \sqrt{\frac{\sum_{b \in \mathcal{T}} \left(r(b) - \hat{r}(b)\right)^2}{|\mathcal{T}|}}, \quad (11)$$

where $\hat{r}(b)$ is the observed success rate of the itemset b in the testing set $\mathcal{T}$. Smaller error-based measures indicate better method performance. We also use standard information retrieval metrics such as Precision, Recall, F1 and Area Under the Curve (AUC). A method that has a higher score is better.

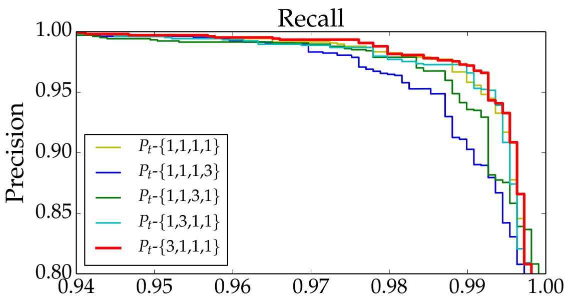
5.3.2 Quantitative analysis. Table 2 presents the performance of all the baseline methods and all the variants of LearnSuc. The table reports MAE, RMSE, AUC and F1 score. Figure 3 presents the precision-recall curves of these methods.

Overall performance. The best baseline method is a heterogeneous network embedding method metapath2vec [11] which gives an MAE of 0.0567 and an RMSE of 0.1648 that are much better than the error by random guess (0.5), because pair-wise similarity plays an important role in generating behaviors. For example, co-authors are often working in very similar research fields. The best variant of our itemset embedding method LearnSuc holds (1) type-distribution-constrained negative behavior sampling strategy and (2) type weights as {3,1,1,1} (authors tend to have higher weights). It gives an MAE of 0.0432 (**-23.8%** relatively) and an RMSE of 0.1243 (**-24.6%** relatively). The traditional dimensionality reduction technique PCA [16] is able to capture 28.4% of total variance. Its performance is not significantly different from the performance of LINE [37]. The network embedding methods show very high AUC and F1 because the pairwise similarities between the items (e.g., authors and keywords) do have impact on the chance of collaboration. Instead, by preserving itemset structures, the itemset embedding method LearnSuc gives **near-perfect** AUC (0.9976, +0.4% relatively) and F1 (0.9847, +1.6% relatively). Figure 3(a) also shows that LearnSuc outperforms the baseline methods. The red curve LearnSuc-P_t is closest to the upper right corner.

Comparing network embedding methods. First, DeepWalk [30] and node2vec [14] perform better than LINE [37] in this task. It shows preserving random walk-based local information or network neighborhoods is more effective than preserving connections and common neighbors on predicting a collaboration. Second, metapath2vec [11] learns the low-dimensional representations of nodes from rich meta path-based features. It models the heterogeneity of the network. It performs the best among the baseline methods in this task. It is important to consider the multiple types of context items in the paper-publishing behaviors.



(a) Compare Itemset Embedding LearnSuc with baseline models.



(b) Compare different settings of context item type weights in LearnSuc-P_t.

Figure 3: Precision-recall curves of the baseline methods and itemset embedding methods (of different settings of item type weights) on context behavior prediction.

Comparing negative behavior sampling strategies. Here we compare LearnSuc-P_t (type-distribution-constrained) with LearnSuc-P_n (size-constrained). The best LearnSuc-P_n sets uniform type weights ({1,1,1,1}) and generates an MAE of 0.0492 (**-13.2%** relatively over metapath2vec) and an RMSE of 0.1439 (**-10.5%** relatively). It shows the effectiveness of itemset embedding on behavior prediction. We observe that LearnSuc-P_t that uses the same uniform type weights can decrease the errors to an MAE of 0.0448 (**-8.9%** relatively over LearnSuc-P_n) and an RMSE of 0.1272 (**-11.6%** relatively). It demonstrates the advantage of type-distribution-constrained sampling – it considers the context types when generating negative samples based on a positive itemset, so the composition of the itemset is carefully modeled. The best LearnSuc-P_t generates **-12.2%** relatively MAE and **-13.6%** relatively RMSE over LearnSuc-P_n.

Comparing different settings on context type weights. In LearnSuc-P_t, we change the weight of one of the context types from 1 to 3 when other weights are controlled as 1. Both Table 2 and Figure 3(b) demonstrate that only the type weights {3 (author),1 (conference),1 (keyword),1 (reference)} can outperform the uniform type weights {1,1,1,1}. The MAE and RMSE are decreased from 0.0448 to 0.0432

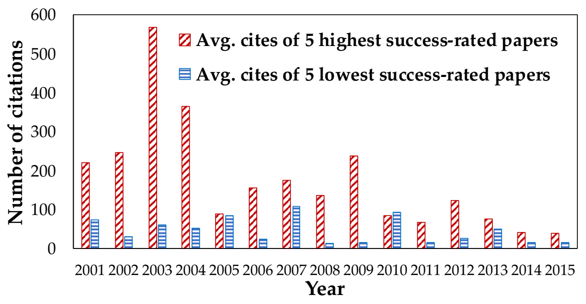


Figure 4: Papers that have higher estimated success rates tend to be cited more (according to Google Scholar Feb'18).

(**-3.6%** relatively) and from 0.1272 to 0.1243 (**-2.3%** relatively), respectively. This matches our intuition: authors play more important roles in making the collaboration successful.

5.3.3 *Qualitative analysis.* We answer three questions here.

Q1: Are papers of high estimated success rates not only successful but also impactful? Yes. We collect two groups of papers in the test set that have the highest/lowest estimated success rate given by LearnSuc. We use Google Scholar to manually find the number of citations of them. Figure 4 presents the average number of citations of those papers published in 2001–2015. By comparing the two paper groups, we observe that the papers of the highest success rate consistently have more citations than those of the lowest success rate. All the papers have been successful (of a success rate above 0.6) but higher success rate indicates more likely to be impactful.

One of the good examples is the paper *"Inferring Social Ties across Heterogeneous Networks"* [35] published in WSDM 2012. The leading author Prof. Jie Tang (Tsinghua University) is a data mining expert on a subset of the keywords such as "factor graph", "heterogeneous network", "predictive model"; and the co-author Prof. Jon Kleinberg (Cornell University) has world-level reputation in computational social science of the keywords such as "social theory", "social influence", and "social ties". The collaboration integrated their complementary expertises, and proposed an effective factor graph based predictive model of inferring social ties across heterogeneous networks. This paper has the highest estimated success rate among the papers in the proceeding. It has been cited over 220 times, tied the best paper and ranked at top 3 among 75!

Q2: What if a negative sample, i.e., a pseudo-paper, has a good success rate? Is it a possibly good paper? Maybe. Most of the pseudo-papers have very small success rate (0.0616 in average), but we do find some of good success rate. The example below has a rate of *0.5491*.

Example 5.1. Authors: Richard Sproat, Weiying Ma, Jiawei Han, Xiaoli Li; *Conference:* SIGIR; *Keywords:* text mining, gaussian processes, biological networks, scalability.

Dr. Richard Sproat studies computational linguistics; Dr. Weiying Ma is an NLP and Information Retrieval scientist; Dr. Jiawei Han is famous for data mining and heterogeneous network mining; Dr. Xiao Li works on bioinformatics and bio network mining. The keywords are quite relevant to their expertises. A paper topic may be reduced as a scalable learning framework for biological text and network mining. However, it is still very difficult to become a real

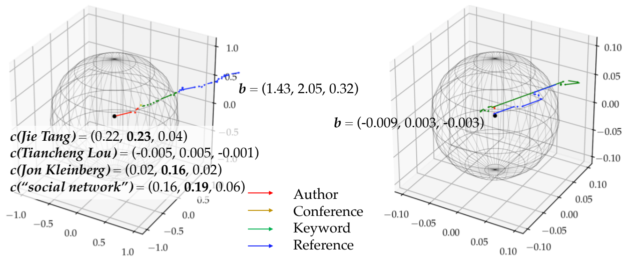


Figure 5: It visualizes a real paper [35] (Left) and a pseudo-paper (Right) in a 3-dimensional embedding space. The item vectors are colored by their context types and learnt by our itemset embedding method LearnSuc-P_t.

paper because LearnSuc does not model the cost of incorporating a new item into an itemset. We leave this issue as a future work.

Q3: Are the negative samples of extremely low success rate likely to be impossible to publish? Yes, much likely. In the pseudo-papers that have very low success rate (below 0.001), we observe that (1) the authors have little chance to build ties between, (2) the authors are not experts in the field of the conference like {"Yan Liu", "SIGGRAPH"}, (3) the keywords are not likely to induce a research topic, such as, {"heterogeneity", "degree of freedom", "biomedical"} and {"citation analysis", "chi squared statistic", "customer retention"}.

5.3.4 *Visualization.* Figure 5 visualizes two itemsets (i.e., papers) in a 3-dimensional embedding space. The left is the WSDM 2012 paper [35] in the above example, and the right is a pseudo paper that was generated by negative behavior sampling. We visualize an itemset with the combination of the vectors of its items. The item vectors are colored by their context types. The paper's vector starts from the original point and consists of the authors' vectors, conference's vector, keywords' vectors and references' vectors.

The vectors of authors *Jie Tang* and *Jon Kleinberg* and the keyword *"social networks"* contribute the most to the magnitude of the real paper's vector. Interestingly, all of them have great scores on the second dimension , which indicates items of different types are well mixed in the embedding space. They collectively elongate the paper's vector significantly towards the dimension. On the other hand, the vectors of items in the pseudo-paper are not always very short in the vector space but they are not complementary with each other. Therefore, the itemset's vector goes back and forth on the dimensions and finally generates a limited distance from the original point.

5.4 Behavioral Context Recommendation

5.4.1 *Validation settings.* For each time of hold out in evaluating the behavior prediction performance, we use the embeddings learnt from the training set and use the testing set to evaluate the recommendation performance. For a test itemset (i.e., paper), given a particular item type (e.g., author, keyword), we hide one of the items of the type in the set. We enumerate every other item of the type in the dataset and use the logistic regression model to predict the success rate of incorporating this extra item into the itemset. We rank the items by the success rate from the highest to the lowest. We assume that the real hidden context item c of the

Table 3: The Harmonic Mean of Ranks (HMR) of network embedding methods and our itemset embedding methods on recommending an item for given itemset. A smaller HMR indicates a better recommendation method.

Method	HMR(author)	HMR(conference)	HMR(keyword)	HMR(reference)
PCA [16]	840.18	4.59	25.75	230.52
LINE [37]	829.06	4.61	27.87	227.24
DeepWalk [30]	796.24	4.32	23.20	100.39
node2vec [14]	660.87	4.41	23.60	89.53
metapath2vec [11]	543.31	4.31	20.43	109.02
LearnSuc-P_n	333.94	4.39	19.59	65.17
LearnSuc-P_t	**266.32**	**4.20**	**18.72**	**64.89**

item type t should be ranked at a higher position if the method makes a better recommendation. We denote $rank(c, b)$ as the rank and use the Harmonic Mean of Ranks (HMR) of all test itemsets to evaluate the performance for each item type:

$$HMR(t) = \frac{|\mathcal{T}^{(+)}|}{\sum_{b \in \mathcal{T}^{(+)}} \frac{1}{rank(c \sim C^{(t)}, b)}}, \quad (12)$$

where $C^{(t)}$ is the set of items of type t and $\mathcal{T}^{(+)}$ is the test set of positive instances. A better method should have a smaller HMR.

5.4.2 Quantitative analysis. Table 3 presents the HMR of network embedding methods and our itemset embedding methods on recommending an item for a given itemset in which we hide an item of a particular type. Overall, our LearnSuc consistently generates lower HMRs on all the context types (author, conference, keyword, and reference) than the baseline embedding methods. With respect to the negative behavior sampling strategies, LearnSuc-P_t outperforms LearnSuc-P_n, showing that type-distribution-constrained negative sampling is effective in learning low-dimensional reprsentations of itemsets for context recommendation.

Recommending a co-author. Among the 12,330 authors in the dataset, given a paper's conference, keywords, references and all the authors except one of them, this author is at the HMR of 266^{th} on the list of authors that our LearnSuc ranks according to the success rate of incorporating him/her into the paper. This is an extremely challenging task. The best baseline metapath2vec ranks the real author at the HMR of 543^{rd}. LearnSuc can find more proper coauthors to make the paper more likely to be published, based on preserving the success structures in the training itemsets.

Recommending a conference. This is not as challenging as co-author recommendation. All the baselines and our methods can rank the real conference at the HMR of 4^{th} to 5^{th} among the 317 conferences.

Recommending a keyword. Our method ranks the real keyword at the HMR of 19^{th} among the 3,548 keywords. The baseline methods have similar performances.

Recommending a reference. This is a challenging task comparable to recommending a co-author. Among the 18,971 references, LearnSuc ranks the real reference at the HMR of 65^{th}, when the best baseline node2vec ranks it at the HMR of 90^{th}. Given authors, keywords and the conference, our method is more likely to recommend proper publications to read and add into the working paper.

Towards real use as recommender systems, the research still has a long way to go but our study presents a novel methodology of multi-type itemset embedding to model the behavior's success.

Efficiency. Our method is implemented on the TensorFlow framework [1]. All methods are parallelized with 24 threads. We utilize the alias sampling method used in LINE. There is no significant difference in running time of LearnSuc. We collected total elapsed time of running on three training sample sizes: 100K (78s), 200K (166s), and 400K (352s). The running time generally grows linearly with the training sample size increasing.

6 CONCLUSIONS

In this paper, we considered behavior as a set of context items and targeted two novel behavior modeling tasks that are (1) predicting the success rate of any set of items and (2) finding complementary items which maximize the probability of success when incorporated into an itemset. We proposed a novel scalable method, multi-type itemset embedding, to learn context item presentations from massive behavior data preserving the success structures. It included a novel measurement of success rate for learning the itemset structures, considered type weights for heterogeneity, and conducted negative behavior sampling for representation learning. We developed the LearnSuc framework and our extensive experimental studies demonstrated its effectiveness and efficiency.

Future work includes various improvements and optimizations. First, as discussed in the Introduction, our LearnSuc framework can be easily applied on various applications with minor adjustment on the formulation of behavior success rate. We essentially look for complementary operators and resources to maximize the probability of achieving a goal. Among many other potential scenarios such as target advertising or military mission planning, predicting success on social media and recommending strategies to improve user's influence seems a natural place to get started. Secondly, underfitting is the major issue of putting itemset embedding into use for massive raw data because of the sparsity of items in the itemsets. For example, suppose we only have author names in the paper's itemset. It doesn't make sense to predict which two or three authors can always publish no matter what problems or methods they propose. Turning unstructured paper text into rich items would be a promising way to alleviate this problem, as well as learning context item type weights automatically. Lastly, updating known item representations with behavior dynamics and inferring unknown new item representation based on current snapshot of the data would be a very interesting yet challenging task. A proper further study would enable the LearnSuc framework on stream data processing and become applicable for more real applications.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. This Research was supported in part by the Army Research Laboratory under the Cooperative Agreement Number W911NF-09-2-0053, and NSF Grants IIS-1447795 and CNS-1622914. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and others. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*, Vol. 16. 265–283.

[2] Deepak Agarwal, Bee-Chung Chen, and Bo Long. 2011. Localized factor models for multi-context recommendation. In *KDD*. 609–617.

[3] Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J Smola. 2014. Cobafi: collaborative bayesian filtering. In *WWW*. 97–108.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, 891–900.

[5] Pablo Castells, Miriam Fernandez, and David Vallet. 2007. An adaptation of the vector-space model for ontology-based information retrieval. *TKDE* 19, 2 (2007).

[6] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *KDD*. 119–128.

[7] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *WSDM*. 295–304.

[8] Weizheng Chen, Xianling Mao, Xiangyu Li, Yan Zhang, and Xiaoming Li. 2017. PNE: Label Embedding Enhanced Network Embedding. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 547–560.

[9] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. *arXiv preprint arXiv:1711.08752* (2017).

[10] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *WWW*. 29–30.

[11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*. 135–144.

[12] Beyza Ermiş, Evrim Acar, and A Taylan Cemgil. 2015. Link prediction in heterogeneous data via generalized coupled tensor factorization. *DMKD* 29, 1 (2015), 203–236.

[13] Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722* (2014).

[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.

[15] Huan Gui, Jialu Liu, Fangbo Tao, Meng Jiang, Brandon Norick, Lance Kaplan, and Jiawei Han. 2017. Embedding Learning with Events in Heterogeneous Information Networks. *TKDE* (2017).

[16] 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.

[17] Yuheng Hu, Fei Wang, and Subbarao Kambhampati. 2013. Listening to the Crowd: Automated Analysis of Events via Aggregated Twitter Sentiment.. In *IJCAI*. 2640–2646.

[18] Zhipeng Huang and Nikos Mamoulis. 2017. Heterogeneous Information Network Embedding for Meta Path based Proximity. *arXiv:1701.05291* (2017).

[19] Mohsen Jamali and Laks Lakshmanan. 2013. HeteroMF: recommendation in heterogeneous information networks using context dependent factor models. In *WWW*. 643–654.

[20] Meng Jiang, Peng Cui, Fei Wang, Xinran Xu, Wenwu Zhu, and Shiqiang Yang. 2014. Fema: flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In *KDD*. 1186–1195.

[21] Meng Jiang, Peng Cui, Nicholas Jing Yuan, Xing Xie, and Shiqiang Yang. 2016. Little Is Much: Bridging Cross-Platform Behaviors through Overlapped Crowds.. In *AAAI*. 13–19.

[22] Meng Jiang, Christos Faloutsos, and Jiawei Han. 2016. Catchtartan: Representing and summarizing dynamic multicontextual behaviors. In *KDD*. 945–954.

[23] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. 1188–1196.

[24] Defu Lian, Zhenyu Zhang, Yong Ge, Fuzheng Zhang, Nicholas Jing Yuan, and Xing Xie. 2016. Regularized Content-Aware Tensor Factorization Meets Temporal-Aware Location Recommendation. In *ICDM*. 1029–1034.

[25] David Matsumoto. 2007. Culture, context, and behavior. *Journal of personality* 75, 6 (2007), 1285–1320.

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

[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[28] Maximilian Nickel and Douwe Kiela. 2017. Poincaré Embeddings for Learning Hierarchical Representations. *arXiv:1705.08039* (2017).

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[31] Ioakeim Perros, Evangelos E Papalexakis, Fei Wang, Richard Vuduc, Elizabeth Searles, Michael Thompson, and Jimeng Sun. 2017. SPARTan: Scalable PARAFAC2 for Large & Sparse Data. In *KDD*.

[32] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*. 693–701.

[33] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*. 81–90.

[34] Alan Said, Shlomo Berkovsky, and Ernesto W De Luca. 2010. Putting things in context: Challenge on context-aware movie recommendation. In *Workshop on Context-Aware Movie Recommendation*. 2–6.

[35] Jie Tang, Tiancheng Lou, and Jon Kleinberg. 2012. Inferring social ties across heterogeneous networks. In *WSDM*. 743–752.

[36] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. 1165–1174.

[37] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.

[38] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. 2009. Dimensionality reduction: a comparative. *J Mach Learn Res* 10 (2009), 66–71.

[39] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *KDD*. 1225–1234.

[40] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *IJCAI*. 2111–2117.

[41] Kai Yang, Xiang Li, Haifeng Liu, Jing Mei, Guo Tong Xie, Junfeng Zhao, Bing Xie, and Fei Wang. 2017. TaGiTeD: Predictive Task Guided Tensor Decomposition for Representation Learning from Electronic Health Records.. In *AAAI*. 2824–2830.