Dynamic Embeddings for Interaction Prediction

Zekarias T. Kefato
zekarias@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Sarunas Girdzijauskas
sarunasg@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Nasrullah Sheikh
nasrullah.sheikh@ibm.com
IBM Research – Almaden
San Jose, USA

Alberto Montresor
alberto.montresor@unitn.it
University of Trento
Trento, Italy

ABSTRACT

In recommender systems (RSs), predicting the next item that a user interacts with is critical for user retention. While the last decade has seen an explosion of RSs aimed at identifying relevant items that match user preferences, there is still a range of aspects that could be considered to further improve their performance. For example, often RSs are centered around the user, who is modeled using her recent sequence of activities. Recent studies, however, have shown the effectiveness of modeling the *mutual* interactions between users and items using separate user and item embeddings.

Building on the success of these studies, we propose a novel method called DeePRed that addresses some of their limitations. In particular, we avoid recursive and costly interactions between consecutive short-term embeddings by using long-term (stationary) embeddings as a proxy. This enable us to train DeePRed using simple mini-batches without the overhead of specialized mini-batches proposed in previous studies. Moreover, DeePRed's effectiveness comes from the aforementioned design and a multi-way attention mechanism that inspects user-item compatibility. Experiments show that DeePRed outperforms the best state-of-the-art approach by at least 14% of Mean Reciprocal Rank (MRR) on next item prediction task, while gaining more than an order of magnitude speedup over the best performing baselines. Although this study is mainly concerned with temporal interaction networks, we also show the power and flexibility of DeePRed by adapting it to the case of static interaction networks, substituting the short- and long-term aspects with local and global ones.

CCS CONCEPTS

• **Information systems** → **Social networks**; **Social recommendation**; • **Computing methodologies** → **Learning latent representations**; **Neural networks**.

KEYWORDS

dynamic embeddings, mutual RNN, recommender systems, interaction prediction, multi-way attention

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450020

ACM Reference Format:
Zekarias T. Kefato, Sarunas Girdzijauskas, Nasrullah Sheikh, and Alberto Montresor. 2021. Dynamic Embeddings for Interaction Prediction. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3450020

1 INTRODUCTION

Vital to the success of a number of real-world recommender systems (RS) is the ability to predict future interactions between entities based on their previous interaction history. In many recommender systems, effective user-item interaction prediction enables end-users to sift through an overwhelming number of choices. In addition, in biology, pharmacology and related fields, interaction prediction between biological and chemical compounds has been explored to better understand unknown bio-chemical interactions [3, 34, 37].

In this paper, we are primarily interested in temporal interaction networks between two sets of entities (*users* and *items*). The terms cover a variety of notions, *e.g.* users could be customers in an e-commerce system, or accounts on Reddit, YouTube or Spotify; items could be products, posts, media produced or consumed by users.

Given a set of observed interactions between users and items, predicting possible future interactions is an increasingly important and challenging task. The goal of this paper is to introduce a new method to predict the next items that users interact with, based on their previous history of interaction. We model our problem through bipartite temporal interaction networks, as they can naturally and effectively represent user-item interactions over time.

Existing studies. In the context of RS, several approaches have been proposed to predict future items a user is likely to interact with, providing encouraging results [4, 5, 10, 16, 24, 29–32]. Often times, however, the focus is on modeling users, while the user-item interaction dynamics that provide a richer signal is overlooked [28]. In several cases, RNNs and other models suitable for sequences were used to train a predictive model over the item sequence corpus.

Recently, studies have shown how to mutually model both user and items based on bipartite interaction networks and demonstrate significant improvement over existing methods [5, 16]. Unlike previous approaches, they have employed mutually recursive RNNs that are more capable to model the user-item interaction dynamics. While they use two types of embeddings, long-term and short-term, the former is just a fixed one-hot vector and the latter is the real

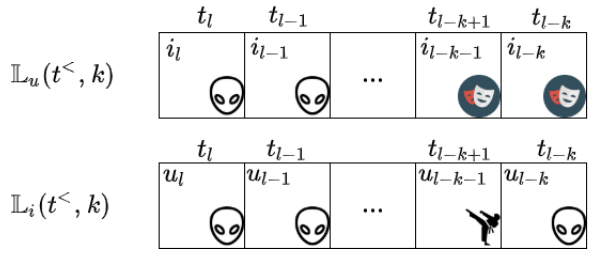

Figure 1: An illustration of participants and the context of the participants of each event for the k recent interaction events of user u ($\mathbb{L}_u(t^<, k)$) and an item i ($\mathbb{L}_i(t^<, k)$). E.g. the last interaction of u was with item i_l from SciFi context (alien icon) and i has interacted with user u_l from SciFi context.

core of their models, that it is used to capture recent user preferences and item properties. Moreover, these approaches work by recursively computing the short-term embedding at time t based on the embedding at time $t-1$, which leads to sequential training that proved to be a bottleneck as the network scales up. Even if recent work has introduced a mini-batch training algorithm, the overhead is not completely alleviated yet [16].

This study. We propose a novel algorithm called DeePRed [1] (Dynamic Embeddings for Interaction Prediction). DeePRed provides a simple yet powerful way of modeling short-term interaction behaviours that removes the aforementioned recursive dependency for efficient training. This is achieved by decoupling the *learnable* user or item embeddings into long-term and short-term embeddings, in order to capture both stationary and transitory interaction patterns. Furthermore, DeePRed computes separate embeddings from the point of view of both: users and items. Henceforth, although our discussion mostly covers users, the same principles can be applied to items unless explicitly stated otherwise.

The key idea behind the effectiveness of DeePRed is that, each time a user interacts with an item, it is modeled using a sequence of k recent items she interacted with, which reflects a context of interaction. For example, Fig. 1 shows the context of the k recent interaction events of a user u and an item i. We see that the two most recent interactions at t_l and t_{l-1} are within the context of SciFi, for both u and i. That is, the last two items that u has interacted with are relevant to the theme of SciFi. Thus, the long-term (contextually stationary) embedding of the items (e.g. $i_l = SpiderMan$ and $i_{l-1} = Terminator$) at t_l and t_{l-1} are used to encode such context.

Similar to previous work [5, 16], we use two mutual RNNs that capture the interaction and temporal patterns within a history sequence (the k most recent interaction events), and generate high-level features. However, unlike previous work, the two RNNs share the same model parameters and are not recursively dependent.

Finally, the power of DeePRed comes from a multi-way attention mechanism that we employ to capture the user-item interaction signal, to check whether the short-term history (k most recent interactions) of a user and an item are compatible using attention weights. The weights are then used as feature selectors over the high-level features and predict the short-term embeddings. In DeePRed, each interaction produces a new instance of short-term embedding for both the user and item. This gives DeePRed the power to reason based on consistent behaviours as opposed to rare events, and it is in contrast to [16] that updates the existing ones. Besides its qualitative power, predicting short-term embeddings as opposed to interaction probabilities is another choice in our design that boosts DeePRed's efficiency.

The last but not the least aspect of DeePRed is that it can be seamlessly extended to tackle static interaction networks. This is achieved by replacing long and short-term aspects with global and local ones, based on a sample of interactions as opposed to the latest (recent) ones.

Our contributions are the following:

- **Novelty**: We propose a novel algorithm that captures user (item) preferences over time by modeling users (items) using their recent interaction history. By leveraging the decoupling of the learnable embeddings, we employ *non-recursive* mutual RNNs to capture interaction and temporal patterns within the histories. Furthermore, an attention mechanism is used to inspect user-item compatibility allowing to significantly improve the predictive performance of our approach.
- **Empirical results:** With respect to the state of the art, our results show at least a 14% gain on mean reciprocal rank, measured on three real-world and publicly available datasets.
- **Efficiency**: As a result of eliminating the recursive self-dependency between short-term embeddings at different time steps, DeePRed achieves more than one order of magnitude speedup over the best performing SOTA methods.
- **Easy extension to static networks**: Though the focus of this study is on temporal interaction networks, we have shown that DeePRed is seamlessly extendable to static interaction networks using three real-world datasets.

2 MODELING PRELIMINARIES

The focus of this study is to model temporal interaction networks; yet, our proposal could be adapted to static networks with little effort. We therefore show first the general model, and then we show how to specialize it for the static case.

We take an ordered set $\mathbb{L}$ containing a log of interactions between a set of users $\mathbb{U}$ and a set of items $\mathbb{I}$, where $L = |\mathbb{L}|$, $U = |\mathbb{U}|$, and $I = |\mathbb{I}|$. An event $e = (u, i, t) \in \mathbb{L}$ records an interaction between a user u and an item i at time t. Events associated with users and items are intrinsically ordered by time. Let $\mathbb{L}_u$ be the set of all interaction events of user u, such that $\mathbb{L}_u = \{e_1, e_2, \ldots, e_l\}$ and events are intrinsically ordered, that is if two events $e_j = (u, i_j, t_j)$ and $e_k = (u, i_k, t_k)$ are such that $j \leq k$, then $t_j \leq t_k$.

In predicting future interactions between users and items, generally, both long-term and short-term interaction behaviours are commonly used [2, 6, 36]. However, short-term behaviours are mostly favored to have a strong impact on follow-up interactions. We adopt a similar assumption and model user and item preferences from both long-term and short-term perspectives. The long-term preferences are captured through the complete interaction histories of users/items. For a user u and an item i, $\mathbb{L}_u$ and $\mathbb{L}_i$ denote their complete interaction history, respectively.

[1] The source code is available at https://github.com/zekarias-tilahun/deepred

Although user preferences are normally considered to change over time [30], we assume that users usually have a dominant (stationary) preference, which remains unchanged. However, as their preferences change over time depending on recent actions, users have a tendency to do related actions. For instance, in movie RS, a particular genre might be preferred by a user at any given time. More importantly, however, one is likely to show interest in movies of different genres based on mood, events in her life (e.g. marriage, childbirth, trauma) and seasons (e.g. Christmas, Summer) [30]. Thus, the most recent watching behaviors have a stronger impact than the old preferences over the next movie that a user is likely to watch.

To capture recent preferences, in line with [2, 35], we use the k most recent interaction events. Unlike some studies [4, 10, 12, 31], however, we assume that the k most recent interaction events from both the user *and* the item influence the next user-item interaction. Later, we shall discuss the details of the benefit of this design choice. Thus, the k most recent interactions of each user $u \in \mathbb{U}$ and item $i \in \mathbb{I}$, respectively, before a given time t are identified by:

$$\mathbb{L}_u(t^<, k) = \{i_j, \Delta_j : (u, i_j, t_j) \in \mathbb{L}_u, t_j < t, j = l-k, \ldots, l\}$$
$$\mathbb{L}_i(t^<, k) = \{u_j, \Delta_j : (u_j, i, t_j) \in \mathbb{L}_i, t_j < t, j = l-k, \ldots, l\}$$

where $\Delta_j = t - t_j$ captures the hotness (recency) of the j^{th} event.

For static networks, we simply strip out time from $\mathbb{L}$; any subsets thereof become unordered. In this case, $\mathbb{L}_u(k) \subseteq \mathbb{L}_u$ and $\mathbb{L}_i(k) \subseteq \mathbb{L}_i$ simply denote a sampled set of k events from observed events $\mathbb{L}_u$ and $\mathbb{L}_i$, respectively.

Research question. The main question of this study is: given an ordered set of observed events $\mathbb{L}_O$, can we design an efficient algorithm that effectively predicts future interactions in temporal interaction networks? In addition, can we make it flexible enough to be applicable to static interaction networks?

3 DEEPRED

The proposed algorithm, DeePRed, captures both stationary and transitory preferences of users and items in interaction networks, by maintaining two dynamic embeddings, one long-term and one short-term, in a latent context-space $\mathcal{S}$. The main hypothesis in DeePRed is that an underlying hidden context-space $\mathcal{S}$ is considered to have been generated as a result of interactions between users and items. This space is assumed to be thematically divided into different regions that are associated to a particular theme or context. For an intuitive understanding of $\mathcal{S}$ in DeePRed, let us consider an illustration shown in Fig. 2. To simplify our discussion, suppose $\mathcal{S}$ is a 2-dimensional euclidean space, which is further divided into three different thematic regions, C_1, C_2, C_3. The notion of a theme/context is related to user interests (preferences) and item properties.

The two dynamic embeddings are updated at every user-item interaction, both for users and items. Since DeePRed applies the same procedure for both, the following discussion is given from a user's perspective. Suppose user u has interacted with an item relevant to context C_2 at time t_1. To reflect such behavior, we start by initializing the user's long-term and short-term embeddings, which are located within the same context C_2.

As time progresses, when the user interacts with different items, new instances of the short-term embeddings are generated by keeping the previous ones. The new instances are shown in the figure along with a timestamp associated to the interaction, which caused the current embedding, and a solid-line trajectory. The motivation for keeping the embeddings comes from a need to maintain a smooth short-term embedding that reflects the "normal" behaviour and the property of a user and an item, respectively. Unless there is a "significant" amount of interactions that cause a drift in a user's interest, for example from C_2 to C_1, "unexpected" interactions should not have a strong influence on future behaviors [15]. Rarely, a user might interact with items from distant contexts (e.g. C_3); for such a temporary case, a new instance can be projected without affecting other short-term embeddings. This allows DeePRed to reason based on embeddings that are closer to a query than exceptional cases. Furthermore, DeePRed gives the flexibility to use embedding histories as needed. In addition, depending on the setting one can choose to discard old embeddings or store them in aggregated form.

The long-term embeddings, on the other hand, are updated and shifted to a new point, discarding the old ones. The dotted line in Fig. 2 shows the trajectory of the long-term embedding of user u, starting from the inactive to active. In a nutshell, these embeddings can be seen as aggregates of the short-term ones over time.

In platforms like YouTube, LastFM, Spotify, the flexibility proposed by DeePRed can be utilized to recommend multi-faceted sets of items, for example one based on short-term embeddings and another based on long-term ones. This is in contrast to several studies that use a single embedding for recommendation.

Modeling long-term interactions. The overall preferences of users and the properties of items are captured by their long-term interactions. To capture patterns in such interactions, we use identity-based embeddings of users and items that live in the same context space $\mathcal{S}$. That is, we use an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{d \times U+I}$, which will be trained as interactions are observed; $\boldsymbol{e}_u$ and $\boldsymbol{e}_i$ denote the long-term embedding of user u and item i. To emphasize that $\boldsymbol{e}_u$ and $\boldsymbol{e}_i$ are conditioned on $\mathbb{L}_u$ and $\mathbb{L}_i$, we use the notation $\boldsymbol{e}_u|\mathbb{L}_u$ and $\boldsymbol{e}_i|\mathbb{L}_i$, respectively.

Modeling short-term interactions. Here the focus is in modeling recent interaction patterns of users and items that govern their follow-up actions. To capture such patterns, we use short-term embeddings $\boldsymbol{u}(t)|\mathbb{L}_u(t^<, k)$ and $\boldsymbol{i}(t)|\mathbb{L}_i(t^<, k)$ for users and items, respectively, conditioned on their recent interactions. In previous studies, short-term embeddings of users and items were recursively dependent on their respective previous short-term embeddings [6, 16]. The recursive nature of these algorithms inherently makes them expensive, as they would need to introduce a specialized algorithm for processing batches of interactions to avoid sequential processing. For example, Kumar et al. had to introduce an algorithm called *t-batch*, that process batches by respecting the temporal order [16]. Our design choice avoids such overhead by relying on the interaction histories rather than the previous short-term embeddings, which allows for "simple" batching.

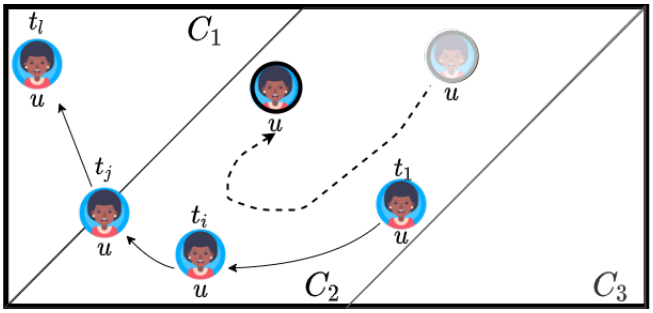

Figure 2: An illustration of the evolution of the short-term and long-term embeddings of user u in a context space, which is further divided into smaller sub-spaces reflecting a context or theme (C_1, C_2, C_3). The dotted arrow indicates the trajectory of the long-term embedding (indicated in black circle). The short-term embeddings of the user are annotated with timestamps, which is associated with the interaction that generated them.

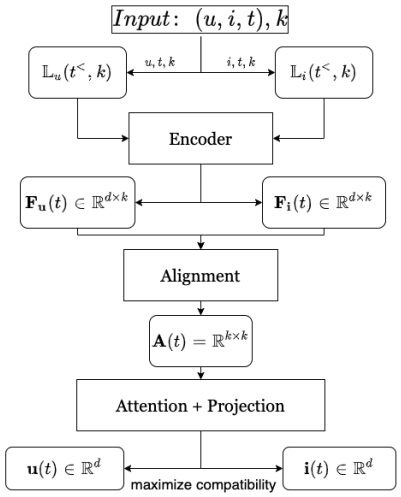

Figure 3: The architecture of DeePRed: Parameters are updated at every event observation (u, i, t). The update is carried out by inspecting the context (captured through long-term embeddings $\boldsymbol{e}_{i_j}$ or $\boldsymbol{e}_{u_j}$, $j = 1, ..., k$) of the k recent events $\mathbb{L}_u(t^<, k)$, $\mathbb{L}_i(t^<, k)$ of u and i that happened just before t. The *Encoder* generates high-level features that capture recurrence and temporal patterns within the k events. The compatibility of each event in $\mathbb{L}_u(t^<, k)$ and $\mathbb{L}_i(t^<, k)$ is quantified using attention weights in the *Attention* phase by leveraging a pair-wise *Alignment* score. Finally, short-term embeddings $\boldsymbol{u}(t)$ and $\boldsymbol{i}(t)$ are obtained in the *Projection* step as weighted sum of the high-level features

3.1 The proposed architecture

The complete architecture of DeePRed is depicted in Fig. 3. The input of DeePRed is given by the observed interaction events and a hyper-parameter k of the model.

Encoder. We process the user and item histories separately, using user and item encoders that share weights. Again, this is in contrast to previous studies that use separate RNN modules that are dependent on previous short-term embeddings. In DeePRed, both the user and item encoder have the same structure; for this reason, most of our discussion is related to the user encoder, while the item encoder is similar. An encoder has two components:

The first component of an encoder computes a signature embedding of the short-term history using the long-term embedding of the items (users) and the deltas as follows:

$$\boldsymbol{S}_u(t) = f(\mathbb{L}_u(t^<, k)) = [[\boldsymbol{e}_{i_j}; \Delta_j] : (i_j, \Delta_j) \in \mathbb{L}_u(t^<, k)] \quad (1)$$

$$\boldsymbol{S}_i(t) = f(\mathbb{L}_i(t^<, k)) = [[\boldsymbol{e}_{u_j}; \Delta_j] : (u_j, \Delta_j) \in \mathbb{L}_i(t^<, k)] \quad (2)$$

The simple, yet expressive and powerful trick used here is that to compute the signature $\boldsymbol{S}_u(t)$ at time t, Eq.(1) relies on the long-term embeddings of the k most recent items that the user u interacted with. Equivalently, in Eq. 2, the k most recent users that interacted with the item i are used to compute $\boldsymbol{S}_i(t)$. The key hypothesis is that the long-term or stationary embeddings of *multiple items* is a strong signal for capturing a user's recent interest, as each stationary embedding $\boldsymbol{e}_{i_j} \in \boldsymbol{S}_u(t)$ captures a sticking property or context (e.g. SciFi) of item i_j. In addition, note that the signature at time t contains information only from the *past* (before time t), as we want to predict the present, time t.

Furthermore, it has been shown that the delay between interactions plays a significant role in predicting future interactions. Thus, each long-term embedding is combined $[\cdot;\cdot]$ with Δ_j in the signature to increase the impact of fresh activities and decrease the importance of the stale ones. Note that, some studies use a decay function for Δ_j instead, e.g. $g(\Delta_j) = 1/\log(e + \Delta_j)$ [2, 35, 36]. In our experiments we found no difference between these approaches, and hence we simply use Δ_j.

The second component of the encoder models recurring interaction and delay patterns in a history using shared and mutual RNN modules over the signatures, $\boldsymbol{S}_u(t)$ and $\boldsymbol{S}_i(t)$. Empirically, Gated Recurrent units (GRU) tend to give better performance, thus we use GRU instead of the basic RNN. Therefore, the standard GRU model for capturing recurrence in a signature $\boldsymbol{S}(t)$ (user or item) slightly modified to integrate Δ_j is given as

$$\boldsymbol{z}_j = \sigma(\boldsymbol{W}_{1z}\boldsymbol{e}_j + \boldsymbol{b}_{1z} + \boldsymbol{W}_{2z}\Delta_j + \boldsymbol{b}_{2z} + \boldsymbol{W}_{3z}\boldsymbol{h}_{j-1} + \boldsymbol{b}_{3z}) \quad (3)$$

$$\boldsymbol{r}_j = \sigma(\boldsymbol{W}_{1r}\boldsymbol{e}_j + \boldsymbol{b}_{1r} + \boldsymbol{W}_{2r}\Delta_j + \boldsymbol{b}_{2r} + \boldsymbol{W}_{3r}\boldsymbol{h}_{j-1} + \boldsymbol{b}_{3r}) \quad (4)$$

$$\boldsymbol{n}_j = \tanh(\boldsymbol{W}_{1n}\boldsymbol{e}_j + \boldsymbol{b}_{1n} + \boldsymbol{W}_{2n}\Delta_j + \boldsymbol{b}_{2n} + \boldsymbol{z}_j \cdot (\boldsymbol{W}_{3n}\boldsymbol{h}_{j-1} + \boldsymbol{b}_{3n})) \quad (5)$$

$$\boldsymbol{h}_j = (1 - \boldsymbol{r}_j) \cdot \boldsymbol{n}_j + \boldsymbol{r}_j \cdot \boldsymbol{h}_{j-1} \quad (6)$$

where σ is the sigmoid function and $\boldsymbol{W}_{pq}$, $\boldsymbol{b}_{pq}$, $p \in \{1, 2, 3\}$ and $q \in \{z, r, n\}$ are the parameters of the model shared by the encoders; $\boldsymbol{e}_j$ corresponds to either $\boldsymbol{e}_{i_j}$ or $\boldsymbol{e}_{u_j}$ depending on the specified signature. At each step j, a new hidden state $\boldsymbol{h}_j$ is computed using the j^{th} step inputs of $\boldsymbol{S}(t)$, *i.e.* the long-term embedding $\boldsymbol{e}_j$ and Δ_j, and the previous hidden state $\boldsymbol{h}_{j-1}$.

Finally, we concatenate the hidden states of the GRU as

$$\boldsymbol{F}(t) = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k] \quad (7)$$

in order to obtain a high-level feature matrix of the signature at time t that captures recurring interaction and delay patterns. Again, depending on the encoder, $\boldsymbol{F}(t)$ is either $\boldsymbol{F}_u(t)$ or $\boldsymbol{F}_i(t)$.

Alignment. Recall that both the user's and item's long-term embeddings live in the same space, and the high-level features $\boldsymbol{F}_u(t)$ and $\boldsymbol{F}_i(t)$ are derived based on such embeddings. Thus, as shown in Eq. 8, the alignment component is used to inspect the compatibility between these features, to see how well the recent events of u and i agree contextually.

$$\boldsymbol{A}(t) = \tanh(\boldsymbol{F}_u(t)^T \boldsymbol{F}_i(t)) \tag{8}$$

We can interpret each row j of $\boldsymbol{A}(t) \in \mathbb{R}^{k \times k}$ as a measure of context agreement between the j^{th} item in the given user's (u) short-term history with all the users in the given item's (i) short-term history at time t. In Eq. 8, similar to [7, 14], one can add more degree of freedom by introducing a trainable parameter $\Theta \in \mathbb{R}^{d \times d}$ depending on the problem setting as in the following equation:

$$\boldsymbol{A}(t) = \tanh(\boldsymbol{F}_u(t)^T \Theta \boldsymbol{F}_i(t)) \tag{9}$$

However, we have empirically observed that for the problem at hand, fixing Θ to the identity matrix $\boldsymbol{I}$ gives a better result. When Eq. 9 is applied, DeePRed tends to overfit faster even with a strong regularization; as a result, we opted for Eq. 8 instead. Hence, the only free parameters of DeePRed are the long-term embedding $\boldsymbol{E}$ and the GRU parameters.

Attention + Projection. Finally, in order to obtain embeddings that reflect short-term behaviours we pay attention to strong contextual agreements in $\boldsymbol{A}(t)$, signaled by high scores, . In other words, we want to investigate the compatibility between the recent interest of a user and the property of an item to understand where the agreement lies. To this end, we compute attention weights for each item in the user's recent history (and vice-versa for each user in the item's recent history) using a column-wise ($\boldsymbol{X}_{\bullet:}$) and row-wise ($\boldsymbol{X}_{:\bullet}$) max-pooling as shown in Eq. 10 and 11, respectively.

$$\tilde{\boldsymbol{u}}(t) = \max \boldsymbol{A}(t)_{\bullet:} \tag{10}$$

$$\tilde{\boldsymbol{i}}(t) = \max \boldsymbol{A}(t)_{:\bullet} \tag{11}$$

The j^{th} component $\tilde{\boldsymbol{u}}_j(t)$ of the vector $\tilde{\boldsymbol{u}}(t) \in \mathbb{R}^k$ corresponds to the attention weight of the j^{th} event, $(i_j, \Delta_j) \in \mathbb{L}_u(t^<, k)$. It indicates:

- the strongest alignment (contextual agreement) of the j^{th} item i_j from all the users in the short-term history $\mathbb{L}_i(t^<, k)$ of the item i
- the hotness of the event

and it is the result of the column-wise pooling on the j^{th} row, $\max(\boldsymbol{A}(t)_{j:})$. These two interpretations of the attention weights are based on the assumption that future activities are governed by recent actions and interest [19, 35, 36]. Inversely, stale events should have less impact on future interactions.

Equivalently, the j^{th} component $\tilde{\boldsymbol{i}}_j(t)$ of $\tilde{\boldsymbol{i}}(t) \in \mathbb{R}^k$ represents the attention weights of the j^{th} event, $(u_j, \Delta_j) \in \mathbb{L}_i(t^<, k)$ and it is the result of the row-wise pooling on the j^{th} column, $\max(\boldsymbol{A}(t)_{:j})$. The interpretation remains the same.

In this way, each item in the user history and each user in the item history are now scored in relation to their contextual agreement, from which we obtain the compatibility between the interacting user and item. Alternatively, we have used mean-pooling in Eq. 10 and 11 and empirically observed no difference.

At this point, we *project* a new point representing the short-term interest and properties using the normalized attention weights. Eq. 12 and 13 compute the user and item projection using the weighted sum of the features $\boldsymbol{F}_u(t)$ and $\boldsymbol{F}_i(t)$, respectively.

$$\boldsymbol{u}(t) = \boldsymbol{F}_u(t) \cdot \texttt{softmax}(\tilde{\boldsymbol{u}}(t)^T) \tag{12}$$

$$\boldsymbol{i}(t) = \boldsymbol{F}_i(t) \cdot \texttt{softmax}(\tilde{\boldsymbol{i}}(t)^T) \tag{13}$$

Both equations can be seen as feature selectors based on contextual agreement and freshness. That is, they select those features that have a strong contextual agreement and are relatively new as indicated by the magnitude of the attention weights. The $\texttt{softmax}(\cdot)$ function gives us a distribution of weights for events in the short-term history of u and i. That is, fresh and contextually agreeing events will get weights close to 1, otherwise close to 0. We argue that the model can learn in a way that weights are distributed in the aforementioned manner. As desired, consequently, weighted features with weights close to 1 will govern the projections. We consider $\boldsymbol{u}(t)$ and $\boldsymbol{i}(t)$ as predictions of the short-term embeddings of the user and item at time t, respectively.

3.2 Training DeePRed

Similarly to previous work [16], DeePRed predicts the user and item embeddings, albeit in a different manner. Thus, we employ a similar loss function using mean squared error. Our goal is to jointly train the long-term and short-term embeddings in order to bring the projection of frequently interacting items as close as possible. To this end, we minimize the L_2 distance as

$$\mathcal{L} = \min \frac{1}{N} \sum_{(u,i,t) \in \mathbb{L}_{train}} ||\boldsymbol{u}(t) - \boldsymbol{i}(t)||_2^2 + \mathcal{L}_{reg} \tag{14}$$

where N is the batch size for batch training and $\mathbb{L}_{train}$ is the observed event log in the training set. The second term on the RHS of Eq. 14, a regularization loss, is introduced to avoid the trivial solution of collapsing into a subspace. It is motivated by the Laplacian eigenmaps method, which adds the constraint $\boldsymbol{u}(t)^T \boldsymbol{i}(t) = 1$ to avoid the collapse. Therefore, we specify $\mathcal{L}_{reg}$ as

$$\mathcal{L}_{reg} = \gamma \cdot ||\boldsymbol{v}^T \boldsymbol{v} - \boldsymbol{I}||_F^2 \tag{15}$$

where $\boldsymbol{v} = [\boldsymbol{u}(t); \boldsymbol{i}(t)] \in \mathbb{R}^{d \times 2}$ and γ is a regularization coefficient. $\mathcal{L}_{reg}$ encourages points to be similar to themselves but not others. Given that we predict embeddings following [1, 16] as opposed to scores as in [6], we do not need for a contrastive loss in Eq. 14.

Since our algorithm is designed in such a way that the short-term embeddings at time t are not dependent on the ones at time $t-1$, batching is straightforward and DeePRed incurs in no overhead from batch processing unlike the work of Kumar et al. [16]. Together with design choices explained above, this makes DeePRed efficient, as demonstrated in Section 4.

3.3 DeePRed for Static Networks

DeePRed requires only minor changes to be applicable to static interaction networks, as explained below.

The first obvious change is the lack of time, and consequently the lack of order; we consider $\mathbb{L}$ to be an unordered set. Thus, the

notion of "long-term" and "short-term" interactions is meaningless. Instead, the equivalent idea in static networks is "global" for "long-term" and "context-aware" for "short-term". Global interactions are modeled as ($\boldsymbol{e}_u|\mathbb{L}_u$ or $\boldsymbol{e}_i|\mathbb{L}_i$) using almost all the observed events in no specific order. We refer to the corresponding embeddings as *global embeddings*. Similarly, context-aware interactions are modeled using *context-aware embeddings* $\boldsymbol{u}|\mathbb{L}_u(k)$ or $\boldsymbol{i}|\mathbb{L}_i(k)$ conditioned on k randomly sampled events. The context-aware embeddings are in line with recent studies that argue against the adequacy of using a single embedding per node [8, 13, 17, 26, 33]. Each node, instead, is represented by multiple embeddings reflecting the multi-dimensional aspect of a node's interest or property.

Thus, the input is specified by each interaction $(u, i) \in \mathbb{L}$ and k. The user and item encoders take $\mathbb{L}_u(k)$ and $\mathbb{L}_i(k)$; encoding amounts to a simple embedding lookup and concatenation operation, to generate $\boldsymbol{F}_u$ and $\boldsymbol{F}_i$ ignoring the GRU model. The followup steps are a straightforward application of the *alignment* first, followed by *attention + projection* to obtain the context-aware embeddings $\boldsymbol{u}$ and $\boldsymbol{i}$.

4 EMPIRICAL EVALUATION

We evaluate the performance of the proposed algorithm using three real-world temporal interaction networks and we compare DeePRed against seven state-of-the-art baselines.

4.1 Datasets

The three publicly available datasets we selected are the following:

- **Reddit** [16] contains post interactions by users on subreddits (items), over a period of one month. The most active users (10,000) and items (1,000) are collected, with 672,447 interactions in total. Actions are repeated 79% of the time.
- **Wikipedia** [16] contains edit interactions by editors (users) on Wikipedia pages (items) over a period of one month. 8,227 editors with at least 5 edits and the 1,000 most edited pages are included, for a total of 157,474 interactions. Actions are repeated 61% of the time.
- **LastFM** [16] contains listening activities by users on songs (items), over a period of one month, restricted to 1,000 users who listened to the 1,000 most-listened songs, with 1,293,103 interactions in total. Actions are repeated 8.6% of the time.

4.2 Baselines

We compare DeePRed with seven state-of-the-art algorithms commonly used in recommender systems, grouped as follows:

- **Sequence models** are different flavors of RNNs trained based on item-sequence data: LSTM, TimeLstm [36], RRN [30], LatentCross [2]
- **Bipartite models** are baselines based on bipartite interaction graph and employ mutually recursive RNNs: DeepCoEvolve [5], Jodie [16].
- **Graph base model**: finally, we have CTDNE-based on continuous-time graph embedding using temporal random walks.

4.3 Next Item Prediction Experiment

Based on observations of recent interactions with items, the goal is to predict the next item a user is likely to interact with. This is what lies at the backbone of a number of RS.

Setting. Similarly to Kumar et al. [16], data is partitioned by respecting the temporal order of events as training (80%), validation (10%), and test (10%). During training, we use the validation set to tune the model hyperparameters using Bayesian optimization.

During testing, given a ground-truth interaction (u, i, t), DeePRed predicts a ranked list of the top-k items that u will interact with at time t, based on previous interactions $\mathbb{L}_u(t^<, k)$ and $\mathbb{L}_i(t^<, k)$. Since DeePRed predicts short-term embeddings, as opposed to interaction probabilities, we can use an efficient nearest-neighbor search to predict the top-k items. We use mean reciprocal rank (MRR) and Recall@k to measure the quality of the ranked list for the top-k items, with $k = 10$ and $k = 1$.

Results. Results are reported in Table 1. Since all the settings are exactly the same, the figures for all the baselines are directly taken from Kumar et al. [16].

DeePRed outperforms all the baselines by a significant margin in all but one case. Almost all the baselines have a huge gap between MRR and Recall@10, unlike the small gap of DeePRed. This shows that DeePRed ranks the ground truth higher, while others simply detect it in lower positions in the top-10 predicted items. For example, for the only case where Jodie beats DeePRed by a small margin, we compare how Jodie and DeePRed exactly match the ground truth, *i.e.*, the Recall@1, and it is 0.648 for Jodie and 0.813 for DeePRed.

We argue that DeePRed's performance is mainly driven by the short-term embeddings projected based on the mutual attention mechanism (compatibility). As this enables DeePRed to project a feature that is contextually related to the recent activities of the users and items, which is widely believed to govern future actions.

Effect of features. One might ask, and rightly so, why not include a richer set of features in DeePRed, as in previous works [2, 5, 16]. First, some of these features (software client, page) are not easily accessible [2]. Other features, such as the textual content, could be easily integrated into our model without affecting the architecture; anyway, we found no difference for the three datasets. To verify this, we have further investigated what happens when you remove textual features from the strongest baseline, Jodie. As shown in Table 2, JodieNF (Jodie with no features) performs as well as Jodie, if not better, for the two datasets with textual interaction features.

4.4 Runtime Experiment

To empirically compare DeePRed's efficiency, we measured the time needed to run the models. In Fig. 4, we report the comparison between the methods for completing an epoch using the Reddit dataset. We see that DeePRed is much faster than all the baselines. Since we are using the figures from [16], Fig. 4 might not be a fair comparison as the machines are different. Hence, we rerun Jodie on our machine and it took 15 minutes to complete the same epoch, showing that the speedup by DeePRed is even better, more than an order of magnitude.

Method	Reddit		Wikipedia		LastFM		Minimum % of improvement of DeePRed over method	
	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10	MRR	Recall@10
LSTM	0.355	0.551	0.329	0.455	0.062	0.119	133.23 %	51.17 %
TimeLstm	0.387	0.573	0.247	0.342	0.068	0.137	113.95 %	45.37 %
RRN	0.603	0.747	0.522	0.617	0.089	0.182	37.13 %	11.51 %
LatentCross	0.421	0.588	0.424	0.481	0.148	0.227	96.67 %	41.67 %
CTDNE	0.165	0.257	0.035	0.056	0.01	0.01	401.81 %	224.12 %
DeepCoEvolve	0.171	0.275	0.515	0.563	0.019	0.039	71.84 %	57.90 %
Jodie	0.726	**0.852**	0.746	0.822	0.195	0.307	14.04 %	-2.23 %
DeePRed	**0.828**	0.833	**0.885**	**0.889**	**0.393**	**0.416**	-	-
% gain over Jodie	14.04 %	-2.23 %	18.63 %	8.15 %	101.53 %	35.50 %	-	-

Table 1: The comparison of the empirical results between DeePRed and the baseline methods for the three temporal datasets. Bold and underline indicate best and second best performing algorithms, respectively.

Method	Reddit		Wikipedia	
	MRR	Recall@10	MRR	Recall@10
Jodie	0.726	0.852	0.746	0.822
JodieNF	0.726	0.852	0.759	0.824

Table 2: Jodie vs JodieNF

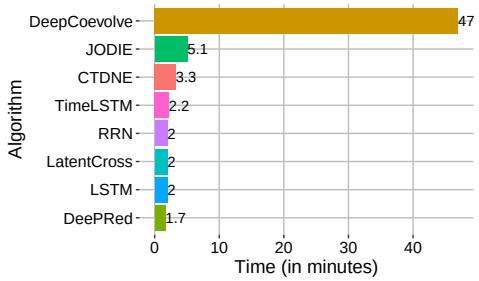

Figure 4: The computational time (in minutes) required to complete an epoch using the Reddit dataset.

4.5 Hyperparameter sensitivity experiment

In this section, we analyze the effect of different hyperparameters of the methods on next item prediction. We simply compare DeePRed with Jodie, since it is much better than all the other baselines.

Impact of training size. Despite their gap, as shown in Fig. 5, 60% of the events are sufficient for both methods to effectively predict the next item on Reddit and Wikipedia. DeePRed executed on LastFM, instead, keeps improving as repeated actions are sparse and patterns might emerge from observing more examples.

Impact of embedding Size. Fig. 6 shows the impact of the embedding size; for DeePRed, 128 is an optimal value, while for Jodie this parameter has almost no influence.

Effect of k. Parameter k, the number of short-term events in $\mathbb{L}_u(t^<, k)$ and $\mathbb{L}_i(t^<, k)$, affects DeePRed only. Our findings are reported in Fig. 7; we observe that k has different effects across datasets. In LastFM, increasing the number of events produces an improvement; in Reddit, there is no effect; in Wikipedia, a declining effect can be observed. Recall that, actions are seldom repeated globally in LastFM, implying that repeated actions are locally sparse; for this reason, interaction patterns are detected by increasing the volume of retrospective observations.

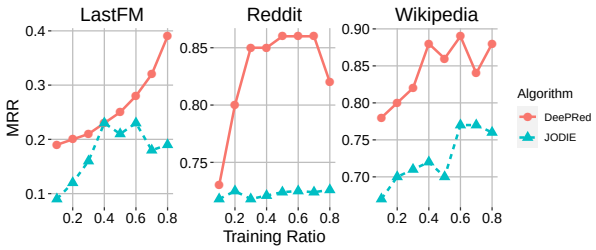

Figure 5: Effect of training proportion

4.6 Static Networks' Experiment

We discuss now our experiments carried out on three static networks. Although DeePRed performs well, our goal here is to show its potential and flexibility, rather than report its superiority.

Datasets. We use the following static interaction networks:

- **MATADOR** (Manually Annotated Targets and Drugs Online Resource) [21] is a drug-target interaction network, with 801 drugs (users) 2,901 targets (items), and 15,843 interactions.
- **SIDER** (Side Effect Resource version 4.1) [18] is a drug (user) and side-effects (item) association dataset. There are 639 users, 10,184 items and 174,977 interactions (associations).
- **STEAM** [22] is a popular PC gaming hub dataset, containing games (items) users have purchased. There are 12,393 users, 5,155 games, and 129,511 purchasing actions.

Baselines. We use four baselines grouped as follows:

- **Context-aware**: Splitter [8] is a SOTA context-aware baseline; similarly to DeePRed, it learns multiple embeddings of nodes for static networks.
- **Context-free** DeepWalk [20], Node2vec [9] and line [25] are popular baselines used for static network embedding

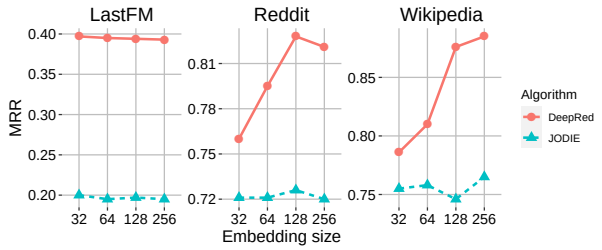

Figure 6: Effect of embedding size

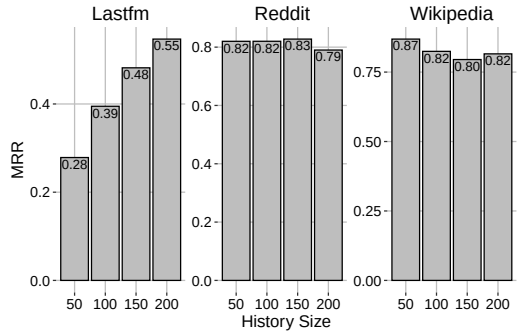

Figure 7: Effect of the short-term history size

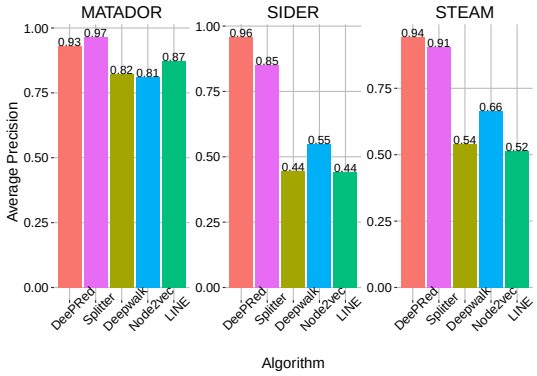

Figure 8: The Average Precision result for interaction prediction on static networks

The interaction prediction is executed as a link prediction task, where we create a random partition of the graph as training (60%), validation (10%), and test (30%) sets. In addition, we randomly sample non-existing (negative) interactions proportional to the test set (30%). An algorithm is trained on the training set and tuned on the validation set. The average precision (AP), which summarizes the precision-recall curve is then computed based on a method's capacity to rank the test (true) over negative (false) interactions. The results reported in Fig. 8 shows that DeePRed is comparable with a context-aware and much better than context-free baselines.

5 RELATED WORK

Factorization methods have significantly influenced the study of recommender systems (RS), more prominently since the Netflix prize competition. However, as deep neural networks (DNNs) gained momentum across several domains, several studies have shown the effectiveness of DNNs in RS as well [4, 11, 28, 30]. Early efforts used a vanilla DNN architecture by integrating crafted and learned features into the models [4].

As recurring patterns in user-item interactions are considered to be critical in recommending or predicting future activities, recurrent neural networks (RNNs) and its variants have been widely used in interaction prediction or RS.

RNNs for Recommender Systems. RNNs are inherently suited for modeling patterns in sequential data, such as language and time-series. Due to their effectiveness, they have seen applicability in different areas, such as NLP, speech recognition, computer vision, and health–just to name a few. Initial efforts in RS have employed RNNs by simply using a sequence of user actions in order to capture repeated user activity patterns, and model their preference or behavior [10, 24, 30]. This approach has further been used to predict interesting items for users based on their preference, for example on platforms like YouTube, Spotify, LastFM. However, standard RNNs and its variants (LSTM, GRU) can only capture recurrence and do not encode delay or interval between activities, which is an intrinsic nature of user behaviours. This is because activities that are close to an event in time are more likely to trigger such event than the ones that are far apart.

Time-based RNNs. Motivated by the aforementioned need, extensions to RNNs (LSTM, GRU) have been introduced to account for time. In addition to the existing gating mechanisms in RNNs, these studies have introduced different time-gating mechanisms to favor new events and discount the impact of old ones [35, 36]. Novelty or oldness refer to the delta in time, not to the position of events in a sequence.

Mutual RNNs. Closely related to our study, recently mutual RNNs for next item prediction have been proposed [5, 16]. A simple yet powerful aspect of these approaches is the bipartite temporal interaction network model, and the mutual RNN architecture that paved a way to examine user-item interaction dynamics. However, besides the essential differences in modeling short-term embeddings of users and items, DeePRed is also different in using shared and non-recursive mutual RNNs.

Other methods. Besides RNNs, other methods such as graph neural networks (GNN) and transformers have also been employed in RS [27]. The former was introduced for neural collaborative-filtering and session-based RS [28, 29, 31, 32]. Due to the ever increasing impact of transformers for modeling sequential data, several studies proposed this model for predicting next basket items [12, 23, 32]. Training transformers has proved to be much more efficient than RNNs, as they are highly parallelizable. However, the core component of transformers–*self-attention*–has the tendency to distribute attention weights and discounting impact from local dependencies [32].

Parameter	Reddit	Wikipedia	LastFM	Matador	Sider	Steam
k	50	50	100	200	200	200
γ	0.6	0.5	0.7	1.0	1.0	1.0
Dropout rate	0.6	0.7	0.8	0.3	0.3	0.3
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Embedding size	128	128	128	128	128	128

Table A1: DeePRed's hyperparameter configurations

6 CONCLUSIONS AND FUTURE WORK

This study presents a novel algorithm called DeePRed for next item prediction in temporal interaction networks. Building up on recent achievements, DeePRed captures the mutual interaction dynamics in the interactions between users and items. We propose a simple yet powerful mechanism to model both user and item short-term preferences based on the their recent interaction history. The history serves as proxy for the context of interaction in recent events. We leverage the mechanism to avoid recursive dependency between consecutive short-term embeddings of a user or an item over time. Our design enables DeePRed to be effective in predicting next item interaction without compromising efficiency.

Our empirical finding on three real-world datasets demonstrate the effectiveness of DeePRed over seven SOTA baselines by at least 14% MRR measure. In addition, DeePRed is at least an order of magnitude faster than the best performing baselines.

We have also shown that the design of DeePRed is flexible enough to accommodate static networks. As a demonstration, we show how well it performs for interaction prediction over biochemical and gaming interaction networks.

Though maintaining multiple embeddings in DeePRed is what lies behind its effectiveness, it comes at the cost of memory. As GPU memory is expensive, this calls for an improved design for DeePRed, that will be addressed in future work.

A DEEPRED CONFIGURATION

Table A1 shows the final configurations of DeePRed used to report the results in Section 4. The experiments are executed on an NVIDIA QUADRO RTX 5000 GPU with NVLink, 3072 CUDA cores, and 16 GB GDDR6 memory.

REFERENCES

[1] M. Belkin and P. Niyogi. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.

[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. ACM, New York, NY, USA, 46–54.

[3] Krisztian Buza and Ladislav Peška. 2017. Drug–target interaction prediction with Bipartite Local Models and hubness-aware regression.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. ACM, New York, NY, USA, 191–198.

[5] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Deep Coevolutionary Network: Embedding User and Item Features for Recommendation. arXiv:1609.03675 [cs.LG]

[6] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. 2016. Recurrent Coevolutionary Latent Feature Processes for Continuous-Time Recommendation. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. ACM, New York, NY, USA, 29–34.

[7] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. arXiv:1602.03609 [cs.CL]

[8] Alessandro Epasto and Bryan Perozzi. 2019. Is a Single Embedding Enough? Learning Node Representations That Capture Multiple Social Contexts. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. ACM, New York, NY, USA, 394–404.

[9] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 855–864.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks.

[11] How Jing and Alexander J. Smola. 2017. Neural Survival Recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 515–524.

[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation.

[13] Zekarias T. Kefato and Sarunas Girdzijauskas. 2020. Gossip and Attend: Context-Sensitive Graph Representation Learning. In *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media, ICWSM 2020, Held Virtually, Original Venue: Atlanta, Georgia, USA, June 8-11, 2020*, Munmun De Choudhury, Rumi Chunara, Aron Culotta, and Brooke Foucault Welles (Eds.). AAAI Press, Atlanta, Georgia, USA, 351–359. https://aaai.org/ojs/index.php/ICWSM/article/view/7305

[14] Zekarias T. Kefato and Sarunas Girdzijauskas. 2020. Graph Neighborhood Attentive Pooling. arXiv:2001.10394 [cs.LG]

[15] Yehuda Koren. 2009. Collaborative Filtering with Temporal Dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France) *(KDD '09)*. Association for Computing Machinery, New York, NY, USA, 447–456.

[16] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, New York, NY, USA, 1269–1278.

[17] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. *Is a Single Vector Enough? Exploring Node Polysemy for Network Embedding*. Association for Computing Machinery, New York, NY, USA, 932–940. https://doi.org/10.1145/3292500.3330967

[18] Kuhn M, Letunic I, Jensen LJ, and Bork P. 2015. The SIDER database of drugs and side effects.

[19] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 969–976. https://doi.org/10.1145/3184558.3191526

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 701–710.

[21] Günther S, Kuhn M, Dunkel M, Campillos M, Senger C, Petsalaki E, Ahmed J, Urdiales EG, Gewiess A, Jensen LJ, Schneider R, Skoblo R, Russell RB, Bourne PE, Bork P, and Preissner R. 2008. SuperTarget and Matador: resources for exploring drug-target relationships.

[22] Steam. 0. https://www.kaggle.com/tamber/steam-video-games.

[23] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1441–1450.

[24] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-Based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) *(DLRS 2016)*. Association for Computing Machinery, New York, NY, USA, 17–22.

[25] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. *LINE: Large-Scale Information Network Embedding.* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1067–1077.

[26] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-Aware Network Embedding for Relation Modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Association for Computational Linguistics, Vancouver, Canada, 1722–1731.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17).* Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[28] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19).* Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/3331184.3331267

[29] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global Context Enhanced Graph Neural Networks for Session-Based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20).* Association for Computing Machinery, New York, NY, USA, 169–178. https://doi.org/10.1145/3397271.3401142

[30] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17).* Association for Computing Machinery, New York, NY, USA, 495–503.

[31] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (Jul 2019), 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[32] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19.* International Joint Conferences on Artificial Intelligence Organization, Macao, China, 3940–3946. https://doi.org/10.24963/ijcai.2019/547

[33] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure Leskovec. 2020. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20).* ACM, New York, NY, USA, 2434–2443. https://doi.org/10.1145/3394486.3403293

[34] Jiaying You, Robert D. McLeod, and Pingzhao Hu. 2019. Predicting drug-target interaction network using deep learning model.

[35] Yuan Zhang, Xi Yang, Julie Ivy, and Min Chi. 2019. ATTAIN: Attention-based Time-Aware LSTM Networks for Disease Progression Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19.* International Joint Conferences on Artificial Intelligence Organization, Macao, China, 4369–4375. https://doi.org/10.24963/ijcai.2019/607

[36] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do next: Modeling User Behaviors by Time-LSTM. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17).* AAAI Press, Melbourne, Australia, 3602–3608.

[37] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. https://doi.org/10.1093/bioinformatics/bty294