ABSTRACT

Modeling a continuous sequence of associations between entities and sets of entities is crucial for domains such as recommender systems, healthcare, education, and communications. However, learning with a sequence of entity-set associations is challenging because of complex entity-entity dependencies due to the associations and the co-occurrence of entities in the sets. Here we propose DES, a model that extends DeepSets to generate dynamic embeddings of entities using a mutually-recursive recurrent model. We create an algorithm Set-AR to effectively predict sets using DES. We show that our model outperforms four state of the art algorithms in predicting the next entity-set interaction by 15.5%.

ACM Reference Format:

1 INTRODUCTION

Continuous sequence of associations between source entities and sets of (potentially different type of) target entities are common in various domains, such as healthcare (a patient exhibiting a set of symptoms in a hospital visit) [17], education (a student taking a set of courses in a given quarter) [15], and recommender systems (a user buying a set of products at a given time) [10], and communication networks (a person sending email to a set of recipients at a given time) [11, 12]. In these cases, the target of each association is a set of entities instead of a single entity [2, 19]. Accurate prediction of the target set in the next association of a given source is a fundamental problem in these domains [2]. For instance, given the past courses a student has taken, it is useful to recommend the next set of courses she should take [6, 13, 22], and predicting the next set of symptoms a patient is likely to exhibit in future visits helps in treatment planning and tracking disease progression [3, 8, 17].

There are two fundamental challenges when working with temporal entity-set associations. The first challenge is to model the temporal evolution of all source and target entities, because the behavior of entities may change over time—while some associations may be permanent, others may be temporary [4, 20, 21]. The second challenge is that the representations of target entities need to learn the entity co-occurrences. For example, some target entities might be substitutable while other entities might be complementary [14]. Both these aspects are essential to learn to make accurate predictions.

Existing works in the literature learn representations from entity-entity associations and are not applicable to entity-set associations [4, 20, 21]. On the other hand, models such as DeepSets learn representations for sets of entities but are not applicable to a sequence of sets [23]. Here we aim to bridge this gap.

Present work. Here we develop DES, a mutually-recursive RNN-based model for learning dynamic representations from a continuous sequence of entity-set associations. DES has two LSTMs to update the embeddings of source and target entities after every interaction—one LSTM is used to update the source entities and another LSTM to update the target entities. Both the LSTMs are interdependent as the output of both the LSTMs serve as inputs to both the LSTMs in the next timesteps (see Figure 1). We use DeepSets to generate the set representations [23].

The LSTM parameters are trained to predict future entity-set interactions. We create an algorithm called Set-AR that allows for effective exploration of the space of possible target sets for efficient set predictions.

Experiments on three real-world datasets show that DES outperforms four state-of-the-art algorithms in predicting future entity-set interactions by 15.5% accuracy.

2 PROPOSED MODEL: DES

Problem Statement: Given a continuous sequence of associations \( \mathcal{A} : \mathcal{A}_j = (s_j, T_j, \tau_j) \) between source entities \( s_j \in \mathcal{S} \) and sets of target entities \( T_j = \{t_1, t_2, \ldots, t_k\} \in \mathcal{T} \) at time \( \tau_j \), we have two goals: (1) learn dynamic representations \( s(\tau_j) \) and \( t(\tau_j) \) for all source and target entities, and (2) design an efficient set search procedure to predict future entity-set interactions.

We first present DES in Section 2.1 to learn dynamic embeddings and then we present the Set-AR algorithm in Section 2.2 to efficiently explore the set space.

2.1 DES: Learning dynamic representations from continuous entity-set associations

An association between a source entity \( s \) and a target set \( T \) of entities reflects new behavior in all the involved entities. Every association generates two dependencies: first, the source and target entities influence one another when they interact, and second, the entities that appear in the target set together influence one another. It is also important to note that the behavior of all entities change over time as their associations with other entities can evolve. Therefore, we create DES that models this co-evolution by jointly learning time-evolving representations \( s(\tau^j) \in \mathbb{R}^n \) and \( t(\tau^j) \in \mathbb{R}^m \) for source entities \( s \in \mathcal{S} \) and target entities \( t \in \mathcal{T} \) at time \( \tau \).

In DES, we split the temporal representation \( s(\tau^j) \) of an entity \( s \) at time \( \tau \) into two components: a long-term component \( s^l(\tau^j) \) that models stable properties of the entity and a short-term component \( s^s(\tau^j) \) that models the recent properties. Similarly, every target representation \( t(\tau^j) \) has a long-term component \( t^l(\tau^j) \) and a short-term component \( t^s(\tau^j) \). Naturally, both the components depend on
We use the cell state of the LSTM as the long-term component of the DES. The embeddings are updated via the target-LSTM using the source embedding and the subset embedding of the rest of the target set entities. For clarity, we only show the update to \( t_1 \), but the same target-LSTM is used to update targets \( t_2 \) and \( t_3 \).

\[
\begin{align*}
\gamma_1 &= \text{Source-LSTM} \\
\gamma_2 &= \text{Target-LSTM}
\end{align*}
\]

Figure 1: DES architecture: Updates are made to the source and target embeddings after source entity \( s \) associates with the target set \( T = \{t_1, t_2, t_3\} \) at time \( \tau \). The source embedding is updated via the source-LSTM using the target set embedding. The target embeddings are updated via the target-LSTM using the source embedding and the subset embedding of the rest of the target set entities. For clarity, we only show the update to \( t_1 \), but the same target-LSTM is used to update targets \( t_2 \) and \( t_3 \).

one another and every new association influences both components, potentially to a different degree.

We create the set representation of \( T = \{t_1, t_2, \ldots \} \) at time \( \tau \) as \( T(T') \) as an aggregate of individual entity representations \( t(t') \tau T^2 \). We use the DeepSets architecture for this aggregation [23].

The embeddings are updated using the following equations after the association between \( s \) and \( T \) at time \( \tau \):

\[
\begin{align*}
s(s^+) &= \gamma_1(s(s^--), T(T'), \Delta_s) \\
t(t^+) &= \gamma_2(t(t^--), s(s^--), T(T' - \{t\} T'), \Delta_t) \forall t \in T
\end{align*}
\]

where \( \gamma_1 \) and \( \gamma_2 \) are trainable transformation functions modeled using an LSTM each, which we explain in detail below. \( s(s^-) \) and \( t(t^-) \) are embeddings of the source \( s \) and target \( t \), respectively, from their previous associations (with any other entity). Further, \( \Delta_s \) and \( \Delta_t \) are the respective values of the elapsed time since their previous interactions. We use it as an additional input in the model as elapsed time has been shown to be predictive in the literature [1, 4, 7, 20, 24]. The DES model is illustrated in Figure 1.

It is important to note the interdependencies in the model: the embedding of the source is updated using the target set and vice versa. Additionally, to update the embedding of a target entity, the embedding of the rest of the entities in the target set are used as inputs too—this is done to ensure that co-occurring entities influence one another [14]. This leads to the mutual coupling between the update functions \( \gamma_1 \) and \( \gamma_2 \).

We use one LSTM each as functions \( \gamma_1 \) and \( \gamma_2 \) [4, 17]. This way all sources share the same LSTM and all targets share the same LSTM. We use the cell state of the LSTM as the long-term component of the embeddings and the hidden state as the short-term components. Together, the complete embedding of each entity is then used to make predictions.

Training the model. We train a prediction function \( \rho \) (in practice, we use a two-layer fully connected network with ReLU) that uses the embeddings \( s(s^-) \) of a source entity \( s \), \( t(t^-) \) of target entity \( t \), and the future time \( \Delta \) to predict if \( s \) and \( t \) will interact at time \( \tau + \Delta \) (where \( \tau + \Delta \) is the time of entity \( s \)'s next interaction). To train the model, we take the true interactions in the training data as the positive examples and we create the negative examples of interaction of the source \( s \) at time \( \tau + \Delta \) using uniform random sampling target entities. \( \rho \) is trained to produce a high score for true associations and low for others. The loss of the predictions are used to update all model parameters.

2.2 Set predictions with Set-AR algorithm

The approach described above to train the model would predict a set of cardinality \( k \) by selecting the top \( k \) scoring entities among all target entities \( t \in T \). However, this naively ignores the entity co-occurrence dependencies between the selected entities because selecting one entity does not affect the selection of other entities [14].

Therefore, to address this challenge, we propose an algorithm to predict sets, Set-AR (short for Add-Remove), shown in the appendix. Set-AR takes the prediction function \( \rho \) as input to score all target entities \( t \in T \). The algorithm iteratively operates in two phases in each iteration—an add phase, followed by a remove phase—till \( k \) target entities have been selected. In the add phase, one target entity that maximizes the score given by \( \rho \) is greedily added to the current selected set of targets. Note that in this phase, \( \rho \) also takes the set representation of the currently selected set as an additional input for predictions. Then in the remove phase, at most one target entity is removed from the selected set. The idea of the remove step is to discard 'incompatible' entities from the set, that were added in preceding iterations.\(^\text{5}\) Thus, at each step, the size of the selected set increases by at most one. This is done till the set has the required \( k \) entities.

Overall, in this section, we presented the DES model that learns dynamic source and target entity representations from a sequence of entity-set associations. We then presented Set-AR that iteratively generates efficient sets for predicting future entity-set associations.

3 EXPERIMENTS

Here we experimentally show the effectiveness of DES to predict the future associations between source and target entities. The setting is the following: given all past associations till time \( \tau \), predict a set of size \( k \) of target entities that a given source entity \( s \) will interact with.\(^\text{5}\) To prevent infinite loops, we do not allow removing the entity \( j \) that was just added (Line 8)
Table 1: Experiment 1: Set Prediction. Jaccard coefficient and Mean Reciprocal Rank (MRR) of the models in predicting the target set that an entity will interact with in the future. N/A indicates that MRR cannot be calculated for Set-AR algorithm as it only returns a set, not a ranking over all candidate target entities.

<table>
<thead>
<tr>
<th>Model</th>
<th>Enron Jaccard</th>
<th>Enron MRR</th>
<th>LastFM Jaccard</th>
<th>LastFM MRR</th>
<th>Instacart Jaccard</th>
<th>Instacart MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepCoevolve [4]</td>
<td>0.174</td>
<td>0.276</td>
<td>0.075</td>
<td>0.178</td>
<td>0.136</td>
<td>0.336</td>
</tr>
<tr>
<td>DeepWalk [18]</td>
<td>0.144</td>
<td>0.255</td>
<td>0.032</td>
<td>0.132</td>
<td>0.091</td>
<td>0.247</td>
</tr>
<tr>
<td>node2vec [5]</td>
<td>0.123</td>
<td>0.241</td>
<td>0.043</td>
<td>0.144</td>
<td>0.092</td>
<td>0.271</td>
</tr>
<tr>
<td>CTDNE [16]</td>
<td>0.171</td>
<td>0.261</td>
<td>0.066</td>
<td>0.162</td>
<td>0.112</td>
<td>0.276</td>
</tr>
<tr>
<td>DES (Proposed method)</td>
<td>0.146</td>
<td>0.326</td>
<td>0.137</td>
<td>0.333</td>
<td>0.144</td>
<td>0.381</td>
</tr>
<tr>
<td>DES + Set-AR (Proposed method)</td>
<td>0.197</td>
<td>N/A</td>
<td>0.143</td>
<td>N/A</td>
<td>0.146</td>
<td>N/A</td>
</tr>
<tr>
<td>Improvement over best baseline</td>
<td>2.3%</td>
<td>5%</td>
<td>6.8%</td>
<td>15.5%</td>
<td>1%</td>
<td>4.5%</td>
</tr>
</tbody>
</table>

at time $\tau + \Delta$. The values $k$ and $\Delta$ are inputs for the prediction as well.

We conduct the experiments on three diverse datasets: communication (Enron email dataset [11, 12]), recommender systems (LastFM dataset [9]), and e-commerce purchases (Instacart dataset [10]).

We compare DES with the following state-of-the-art algorithms: (a) DeepCoevolve [4, 21] (the closest algorithm), (b) DeepWalk [18], (c) node2vec [5], and (d) CTDNE [16].

**Evaluation metrics:** We compare the models using two metrics: the mean reciprocal rank (MRR) and Jaccard coefficient between the predicted set and ground truth set. MRR is the standard metric used in recommender systems and we use Jaccard coefficient due to the set setting. As Set-AR algorithm only returns a set, we only report the Jaccard coefficient for it.

**Experimental setup:** We train all models using the first 5000 interactions, validate using the next 1000, and test using the next 1000 interactions.

**Model details:** We implemented the model in PyTorch with the Adam optimizer. We trained the model with multilabel margin loss and backpropagated the loss after every 100 interactions. We train the model for 50 epochs. All embeddings were of size 50.

### 3.1 Experiment 1: Set Prediction

In this experiment, we use the embedding of the source entity $s$ and target entity $t$ at time $\tau$ to predict the target set that $s$ will interact with at time $\tau + \Delta$. The predicted and the ground-truth sets are then used to calculate the Jaccard coefficient, and the prediction scores on all candidate target entities is used to calculate the mean reciprocal rank. The results for all the models is shown in Table 1. We see that DES, along with the Set-AR algorithm has the highest Jaccard score in all three datasets (mean improvement of 10% over the best baseline), and DES has the highest MRR score for all datasets (mean improvement of 7% over the best baseline). This shows that DES and Set-AR are able to predict future entity-set associations with significantly better performance than existing models. Moreover, we observe that DES with Set-AR performs better than DES alone, showing that the greedy algorithm is indeed able to lead the model to explore a large set space for predictions.

### 3.2 Experiment 2: Run-time analysis

We conducted an experiment to measure the time taken by the models to train on the Enron dataset. We report the average training time (in seconds) per epoch. For comparison, we also computed the running time of DeepCoevolve, the best-performing baseline.

We find that DeepCoevolve takes 171.37 seconds/epoch, DES takes 183.65 seconds/epoch, and DES with Set-AR takes 1699.86 seconds/epoch. While DES has comparable running time as DeepCoevolve, the training time of our model increases when using the Set-AR algorithm, because of the multiple set operations that it requires. Thus, using Set-AR improves the performance but uses 10 times more running time.

### 3.3 Experiment 3: Contribution of long-term and short-term components

Here we compare the performance when using only the long-term or the short-term component in DES for set prediction. We conduct this experiment on the Enron dataset. Using only the long-term component for prediction, DES achieves a Jaccard coefficient of 0.177, which is 2% less than using both components ($\Delta = 0.197$ Jaccard coefficient). On the other hand, using only the short-term component of DES gives 0.181 Jaccard coefficient, an 1.6% reduction in performance compared to using both components. It is interesting to note that the short-term component leads to slightly better performance than using the long-term component.

Overall, by conducting experiments on three different datasets, we illustrate that DES learns efficient representations of entities given a continuous sequence of entity-set associations. We show that DES with Set-AR is accurate in predicting future sets, though with increased runtime cost.

### 4 CONCLUSIONS

In this paper, we developed DES to learn dynamic embeddings for entities from a continuous sequence of entity-set associations. We further created Set-AR to explore several target sets to select the most likely one for future associations. We experimentally showed that the learned embeddings are useful to make accurate predictions of future entity-set associations.

### REFERENCES


This appendix shows the pseudo-code of the proposed Set-AR algorithm.

**Algorithm 1: Set-AR Algorithm**

**Input**: Source entity representations $s(s)$, target entity representations $t(t) \forall t \in T$, current time $\tau$, time difference $\Delta$, scoring function $\rho$, output set cardinality $k$, set embedding function $T$

**Output**: Predicted set of target entities $R$ : $|R| = k$ that the source entity $s$ will be associated with at time $\tau + \Delta$.

1. $R \leftarrow \emptyset$
2. **while** $|R| < k$ **do**
   3. /* Add phase: select a target entity to be added to the current set $R$ */
   4. **for** $t \in T$ **do**
   5.   $score[t] \leftarrow \rho(T(R \cup \{t\}), s(s), \Delta)$;
   6. **end**
   7. $t^* \leftarrow \argmax score[t]$;
   8. $R \leftarrow R \cup \{t^*\}$; // Add the highest scoring target entity to the current set $R$
   9. /* Remove phase: Select at most one target entity to be removed from $R$ */
   10. **for** $t \in (R - \{t^*\} \cup \{\})$ **do**
    11.   $rscore[t] \leftarrow \rho(T(R - \{t\}), s(s), \Delta)$;
    12. **end**
    13. $t^* \leftarrow \argmax rscore[t]$;
    14. $R \leftarrow R - \{t^*\}$; // Remove the least compatible target entity $t^*$ from the current set $R$
15. **end**
16. **return** $R$