Bipartite Dynamic Representations for Abuse Detection
Abusive Actors Harm Online Communities

Social and collaboration networks:
Trolling, propagating misinformation, offensive language

E-commerce websites:
Fake reviews or purchases to inflate product rankings

Image source: Wikimedia Commons
Abusive Actors Harm Online Communities

Effects of abuse:
• Hurts the experience of honest users
• Reduces customer engagement and revenue
• Reduces trust in the platform

Image source: Wikimedia Commons
Abuse Detection in E-commerce

- Abuse detection is an important problem in the E-commerce setting, where users and items form a (bipartite) interaction graph.
Problem Setting: Detecting Abusive Actors

- **Given:**
  - Interaction data of users with items (e.g. products, subcommunities) over time
  - A subset of users labeled as abusive or non-abusive

- **Goal:**
  - For all unlabeled users, predict if each is abusive
Abuse Detection is Challenging

Complex dynamics: Users interact with various items over time

Extremely large-scale interaction data

Collecting ground-truth labels is costly
Abuse Detection: Interactions Follow Complex Dynamics

1. **Complex dynamics**: users interact with various items over time

**Time domain**: abusive users (red) have more fluctuating activity than normal users (blue)

**Graph domain**: abusive users have less likely connections than normal users (as estimated by link prediction model)
2. Extremely large-scale interaction data

- High volume (100M+) of interactions
- End-to-end training of graph neural networks (GNNs) is expensive:
  - The entire graph and features cannot be fit into GPU memory
  - Exponential memory cost $O(d^L)$ of minibatch training with increasing number of hops $L$, and fanout $d$
- Require a dynamic model

Abuse Detection: Scalability is Crucial
Abuse Detection: Labeled Data is Sparse

3. Collecting ground-truth labels is costly

• Human annotation is expensive

• Positive labels are scarce due to rarity of abuse

Our work: bridge the gap via a comprehensive system for large-scale abuse detection on dynamic graphs
Our Work: Problem Framework

• Model data as a **dynamic, bipartite graph** $G = (U, V, E, f, g, h)$
  • **User** nodes $u \in U$, item nodes $v \in V$
  • Dynamic edges $(u, v, t) \in E (u \in U, v \in V, t \in \mathbb{R})$ with timestamp $t$
  • **Node features** $f: U \rightarrow \mathbb{R}^{d_1}$, $g: V \rightarrow \mathbb{R}^{d_2}$
  • **Edge features** $h: E \rightarrow \mathbb{R}^{d_3}$

• Model learning problem as **transductive node classification**
  • Some users are known to be abusive or non-abusive
    • Partial function $f: U \rightarrow \{0,1\}$ known at training time
  • Goal: infer status of remaining users
    • Predict labels of all users $u \in U$
Abuse Detection is Challenging

Complex dynamics: Users interact with various items over time

Extremely large-scale interaction data

Collecting ground-truth labels is costly
Our Work: Detecting Abuse

Complex dynamics: Users interact with various items over time

Extremely large-scale interaction data

Collecting ground-truth labels is costly

Our approach: BiDyn

Joint modeling of time and graph information

Scalable training scheme

Self-supervised pretraining framework

Model architecture

Training algorithm

Training objective
Our Work: Detecting Abuse

Complex dynamics: Users interact with various items over time

Extremely large-scale interaction data

Collecting ground-truth labels is costly

Our approach: BiDyn

Joint modeling of time and graph information

Scalable training scheme

Self-supervised pretraining framework

Model architecture

Training algorithm

Training objective
BiDyn Architecture: Joint Modeling of Graph and Time Information

- Motivation: preserve raw time series information (bursty behavior)
- BiDyn uses a three-phase model architecture:

1. RNN phase
2. GNN phase
3. Prediction

RNN

GNN

Users → Items

Output head → Abuse probability

Abuse probability

September 16, 2021
Rex Ying, Stanford University
BiDyn Architecture: RNN Phase

- RNN: LSTM of all events associated with given node, in time order
- Aggregate simultaneous events using a `deepset` encoder
- Concatenate number of events on each day
Graph neural network: aggregate information from local neighborhood

- Use convolution function without learnable parameters for aggregation

Obtain the embedding of the user

September 16, 2021
Rex Ying, Stanford University
Our Work: Detecting Abuse

Complex dynamics: Users interact with various items over time

Extremely large-scale interaction data

Collecting ground-truth labels is costly

Our approach: BiDyn

Joint modeling of time and graph information

Scalable training scheme

Self-supervised pretraining framework

Model architecture

Training algorithm

Training objective
BiDyn: Scalable Training Scheme

- Idea: *stacked* ensemble model
- Evolve node representations by alternating application of RNN phase ("**user round**") and GNN phase ("**item round**")
BiDyn: Alternating Training

- **User round**: update user representations by aggregating representations of all items each user interacted with over time
  - RNN during user round ensures accurate modeling of bursty behavior

\[
\begin{align*}
\text{Users} & \quad \text{Items} \\
\text{RNN} & = \quad 1. \quad \bullet \\
& \quad 2. \text{Using} \quad \bullet, \text{predict abuse label and backpropagate}
\end{align*}
\]

...repeat for all users
**BiDyn: Alternating Training**

- **Item round**: update item representations by aggregating representations of all users that interacted with each item

![Diagram showing the BiDyn model](image)

- Graph convolution layer
- ...repeat for all items
BiDyn: Alternating Training

- Improved user representations give rise to improved item representations, and vice versa
- Alternating user and item updates ensure that each node can communicate with every other node in the graph
Our Work: Detecting Abuse

Complex dynamics: Users interact with various items over time

Extremely large-scale interaction data

Collecting ground-truth labels is costly

Our approach: BiDyn

Joint modeling of time and graph information

Scalable training scheme

Self-supervised pretraining framework

Model architecture

Training algorithm

Pre-training framework
Experiments: Datasets

- **E-commerce**
  - Purchases between **buyers** and **products** over two-month period
  - 400K users (0.2% abusive), 5.6M items, 113M edges

- **Wikipedia**
  - Edits between **users** and **articles**
  - 8K users (2.3% abusive), 1K items, 157K edges

- **Reddit**
  - Posts by **users** in **communities**
  - 10K users (3.3% abusive), 1K items, 672K edges
Experiments: Method Comparison

Baselines

- **TGAT**: temporal attention layers to attend to node history
- **TGN**: uses memory modules and graph-based operators
- **DyRep**: model dynamics jointly on small and large time scales
- **JODIE**: predict trajectories of node embeddings over time

Ablations

- **RNN**: use only the RNN phase of BiDyn
- **GNN**: use only the GNN phase of BiDyn
- **RNN-GNN**: use the RNN and GNN phases together, but train end-to-end
### Experiments: Method Comparison

<table>
<thead>
<tr>
<th>AUROC</th>
<th>e-commerce</th>
<th>Wikipedia</th>
<th>Reddit</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGAT</td>
<td>-4.0</td>
<td>73.6</td>
<td>51.5</td>
</tr>
<tr>
<td>TGN</td>
<td>OOM</td>
<td>49.0</td>
<td>67.0</td>
</tr>
<tr>
<td>DyRep</td>
<td>OOM</td>
<td>52.5</td>
<td>61.4</td>
</tr>
<tr>
<td>JODIE</td>
<td>OOM</td>
<td>53.0</td>
<td>61.2</td>
</tr>
<tr>
<td>BiDyn</td>
<td>+1.2</td>
<td>86.5</td>
<td>56.0</td>
</tr>
<tr>
<td>BiDyn + pretraining</td>
<td>+4.5</td>
<td>87.5</td>
<td>50.5</td>
</tr>
</tbody>
</table>

- BiDyn achieves comparable or better performance than existing dynamic graph models on small datasets (Wikipedia, Reddit).
- BiDyn scales to 100M-edge graphs (e-commerce), while baselines run out of memory (OOM).

5% of labels seen in training.
Experiments: Method Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>e-commerce</th>
<th>Wikipedia</th>
<th>Reddit</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNN</td>
<td>+0.0</td>
<td>69.6</td>
<td>53.7</td>
</tr>
<tr>
<td>RNN</td>
<td>+0.0</td>
<td>78.0</td>
<td>51.3</td>
</tr>
<tr>
<td>RNN-GNN</td>
<td>-2.2</td>
<td>70.0</td>
<td>53.8</td>
</tr>
<tr>
<td>BiDyn</td>
<td>+1.2</td>
<td>86.5</td>
<td>56.0</td>
</tr>
<tr>
<td>BiDyn + pretraining</td>
<td>+4.5</td>
<td>87.5</td>
<td>50.5</td>
</tr>
</tbody>
</table>

- BiDyn outperforms models that only use graph or time information alone (RNN, GNN)
- BiDyn outperforms end-to-end training (RNN-GNN) due to increased model depth
- Pretraining can lead to additional gains in label-sparse settings
BiDyn uses 10-100X less memory than baselines, with comparable runtime.

- Enables training of deeper models.
Experiments: Pretraining

- BiDyn is robust to sparse training labels
- Pretraining gives initial separation of node classes
Conclusion

- Integrating time series and graph information leads to significant improvement in abuse detection
- Alternating training is a high-performing, scalable approach to node classification on dynamic graphs
- Pretraining on anomaly detection objectives can improve performance in label-sparse settings
References

- **Inductive representation learning on temporal graphs.** Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, Kannan Achan. ICLR 2020.
- **Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks.** Srijan Kumar, Xikun Zhang, Jure Leskovec. KDD 2019.
- **Temporal Graph Networks for Deep Learning on Dynamic Graphs.** Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, Michael Bronstein. ICML Workshop on Graph Representation Learning 2020.
- **Simplifying Graph Convolutional Networks.** Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, Kilian Q. Weinberger. ICML 2019.
- **Predict then Propagate: Graph Neural Networks meet Personalized PageRank.** Johannes Klicpera, Aleksandar Bojchevski, Stephan Günnemann. ICLR 2019.

- Code: [http://snap.stanford.edu/bidyn](http://snap.stanford.edu/bidyn)