When to Remember Where You Came from: Node Representation Learning in Higher-order Networks

Caleb Belth
University of Michigan
cbelth@umich.edu

Fahad Kamran
University of Michigan
fhdkmrn@umich.edu

Donna Tjandra
University of Michigan
dotjandr@umich.edu

Danai Koutra
University of Michigan
dkoutra@umich.edu

ABSTRACT

For trajectory data (e.g., flight itineraries) that tend to have beyond first-order (i.e., non-Markovian) dependencies, higher-order networks have been shown to accurately capture details lost with a standard (aggregate) network representation. At the same time, representation learning has shown success on a wide range of network tasks, removing the need to hand-craft features for these tasks. In this work, we propose a node representation learning framework called EVO or *Embedding Variable Orders*, which captures non-Markovian dependencies by combining work on higher-order networks with work on node embeddings. We show that EVO outperforms baselines in tasks where high-order dependencies are likely to matter, demonstrating the benefits of considering high-order dependencies in node embeddings. We also provide insights into when it does or does not help to capture these dependencies. To the best of our knowledge, this is the first work on representation learning for higher-order networks.

ACM Reference Format:
Caleb Belth, Fahad Kamran, Donna Tjandra, and Danai Koutra. 2019. When to Remember Where You Came from: Node Representation Learning in Higher-order Networks. In *Proceedings of MIG 2019.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recent work on higher-order networks[1] (HONs) has demonstrated the importance of considering non-Markovian dependencies when building a network representation from trajectory data [12, 13]. Meanwhile, representation learning has been useful for learning feature representations for standard (first-order) networks that can be directly used for downstream tasks [14]. While high-order dependencies are important for accurately *modeling* a network, it is not immediately obvious that the added modeling accuracy is *always* useful to learn over. To understand when it is useful, we evaluate the performance of learning over HONs on several classification and clustering tasks, and compare to the performance of learning over standard network representations.

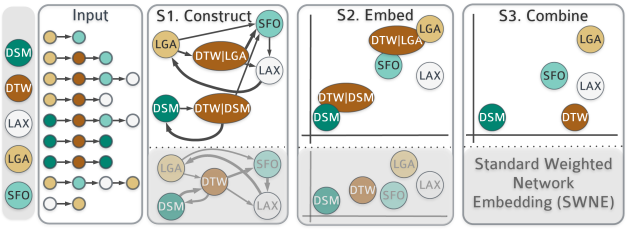

Figure 1: The proposed EVO framework (white background), contrasted with the standard approach SWNE (gray background) (§ 4.2). EVO takes trajectories as input (e.g., flight itineraries) and, in step 1, creates a HON that may contain multiple versions of a node (e.g., DTW is mapped to two conditional nodes, DTW|LGA and DTW|DSM, since the 0.5 probability of visiting SFO from DTW changes to 0.67 or 0.33 depending on how DTW was reached). In step 2, EVO learns node embeddings from the HON, and in step 3, it combines the embeddings of all node versions into a single embedding that captures the important properties of *all* its versions. Note that SWNE has only two steps.

We focus on two research questions: (RQ1) how can we capture non-Markovian dependencies in feature representations and (RQ2) when are these dependencies useful? In response to the first question, we propose EVO (Figure 1), a modular framework for representation learning that works with HON representations to learn dependency-preserving node representations. In response to the second question, we find that EVO is effective at capturing higher-order dependencies and is useful for a range of tasks related to network trajectories. However, contrary to what might be expected, we find that capturing high-order dependencies for tasks where they are not very relevant is not only ineffective, but can be *detrimental*, suggesting that capturing more information is not always helpful. This is a significant finding, since HON representations consist of multiple nodes for multiple dependencies, requiring extra computational power—something highly undesirable for large graphs. Our main contributions are:

- We propose a **general, modular framework** that combines work in representation learning and HONs to generate node embeddings that capture non-Markovian dependencies. It readily

[1]Note that we are using "higher-order" to refer to *non-Markovian* or *beyond first-order* dependencies in trajectories, rather than motifs [10]

MIG 2019, Aug 5, Anchorage

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Major symbols and their meanings.

Symbol	Definition
V, E	a set of nodes and edges, resp.
T	trajectories over nodes V
$G(V, E)$	standard network
$H(V_H, E_H, \delta)$	HON w/ mapping from V to V_H
$\mathbf{r}_v$	embedding of node v

supports both neighborhood and structural embeddings, and works with any embedding approach.
- Through diverse experiments, we bring insights into **when downstream tasks can benefit** from these embeddings.

2 RELATED WORK

Higher-order Networks attempt to preserve non-1st-order dependencies in a network representation. While works such as [11] used a fixed-order representation, generally only modeling 2nd-order dependencies, recent work developed network representations with variable-order dependencies, which are automatically identified from trajectories [12, 13]. Xu *et al.* [13] build a variable-order network by creating multiple versions of nodes to capture multiple dependencies. The resulting HON admits standard graph methods. Another approach [12] generates a layered network, each layer corresponding to a progressively higher dependency level. Since the network is multi-layered, it may require tailored graph methods.

Representation Learning aims to learn feature representations for nodes (or other structures) which can be directly used for network tasks. Many methods are inspired by word embeddings [6], replacing the linear context around words with graph structure around nodes by using variants of random walks. Some are proximity-based, e.g. [3, 7], while others are structural, e.g. [9]. Other methods, e.g. [5], utilize the connection between matrix factorization and skip-gram with negative sampling [8]. For more, [14] surveys the extensive work in representation learning. To the best of our knowledge, we are the first to look at representation learning for HONs. While [10] uses similar terminology, it is not closely related, using "higher-order networks" to refer to graph motifs, whereas we use it to refer to non-Markovian dependencies.

3 METHOD

3.1 Preliminary Definitions

A (standard or first-order) *network* or *graph* $G = (V, E)$ is a set of nodes and a set of edges among them, optionally with weights denoting connection strengths. G implicitly encodes the Markov assumption.

We follow [13] and define a *higher-order network (HON)* $H = (V_H, E_H, \delta)$ as a network, along with a bijective function $\delta : V \rightarrow \mathcal{P}(V_H)$ mapping each node $v \in V$ to the subset of nodes in V_H encoding v's dependencies. For instance, in Figure 1, δ(DTW) = {DTW|LGA, DTW|DSM}. Its inverse maps the nodes back: δ^{-1}({DTW|LGA, DTW|DSM}) = DTW.

Table 2: Dataset statistics: The median length of trajectories, the max dependency found by [13], the size of G (formed by aggregating the edges appearing in T), and the size of H, respectively.

Dataset	Med. Traj. Len.	Max Dep.	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert V_H\rvert$	$\lvert E_H\rvert$
Us Flights	5	4	175	1,598	9,776	49,700
London Tube	12	2	268	646	1,029	2,073

3.2 Problem Definition

Problem 1. *Given a set of trajectories T over nodes V, where the Markov assumption may be violated, find a representation $\mathbf{r}_v \in \mathbb{R}^d$ for each node $v \in V$ such that the representations capture the non-Markovian dependencies.*

3.3 EVO: Embedding Variable Orders

EVO or "Embedding Variable Orders" works in three steps:

(S1) Construct Higher-order Network. Any approach for building a HON (e.g. [13]) can be used as long as there exists some δ and δ^{-1} to map nodes to and from V_H.

(S2) Learn Representations from HON. For each node $h \in V_H$, we obtain its representation $\mathbf{r}_h$. Assuming that the HON is a conventional graph (e.g., not layered), any representation learning method can be used.

(S3) Combine Representations. This step uses a function $f : \mathcal{P}(\mathbb{R}^d) \rightarrow \mathbb{R}^d$ to create a single representation for each node in V from its corresponding nodes in V_H. For any node v, we get its embedding (denoted $\mathbf{r}_v$) by applying f (e.g., average, max, etc.) to the embeddings of all the nodes in V_H which correspond to v (which may vary in number per node). Let $R = \{\mathbf{r}_h \ : \ h \in \delta(v)\}$ be the set of embeddings of nodes in $\delta(v)$, which we acquired in step 2. Formally, $\mathbf{r}_v = f(R)\ \forall v \in V$.

In summary, the EVO framework allows multiple methods, each requiring the selection of (1) a HON representation, (2) a representation learning method, and (3) a function f to convert the representations of HON nodes V_H to representations for nodes V.

4 EVALUATION

In this section, we discuss our experiments, which aim to answer two main research questions:

- **RQ1 Performance:** Does EVO effectively capture high-order dependencies?
- **RQ2 Appropriate Uses:** In what situations does it make sense to capture high-order dependencies? We consider a variety of node classification and clustering tasks.

4.1 Datasets

We use two datasets [12] (Table 2): (1) A London Tube dataset containing passenger itineraries from the London subway system and (2) a US Flights dataset containing flight itineraries, such as LGA → DTW → SFO (Figure 1), meaning someone flew from LaGuardia to Detroit to San Francisco.

4.2 Baselines

We consider three baselines:

- SNE or "Standard Network Embedding" aggregates the trajectories into a standard network representation, where an edge exists between two nodes if the edge occurred in at least one trajectory. Node representations are learned via the same embedding approach used in EVO.
- SWNE or "Standard Weighted Network Embedding" is identical to SNE except that it introduces edge weights, which count the edge occurrences in the trajectories.
- ARWE or "Altered-Random-Walk Embedding" is an approach we introduce. ARWE runs variable-order Markov random walks to generate context for skip-gram-based representation learning directly, *without* building the intermediate HON. The random walks simulate trajectories by obeying the same dependencies used to build H.

4.3 Experimental Setup

For **S1** of EVO, we construct a HON using [13] without using the optional min-support parameter. For **S2**, we use node2vec [3], struc2vec [9], or xNetMF [5], depending on the task. xNetMF and struc2vec do not use weights. Furthermore, xNetMF does not use random walks, while struc2vec's walks are on a k-level graph, so SWNE and ARWE are not usable here. For node2vec and struc2vec, we set the walk length to be the median trajectory length for that dataset (e.g. 5 for Flights), to treat random walks as simulations of real trajectories. We perform 50 walks per node, and use a window-size of 10. We use none of struc2vec's computational optimizations. For **S3**, we consider multiple options—max, min, average, and Hadmard product—for f, each of which is applied elementwise. We also tried concatenation, but saw similar or worse results. We found that max and average worked best, so we report results for only these, named EVO-max and EVO-avg.

Results are reported as averages with standard deviations over 10 runs. The top result for each metric is shaded gray. If the result is statistically significant (which we consider $p \leq 0.05$ for a paired t-test), we mark it with an "*". We discuss results in each task's section, and takeaways in § 4.6.

4.4 Node Classification

We designed four binary classification tasks to answer our research questions. We use node representations of each method as input to a random forest classifier with 100 trees, running with 10 different 70%/30% train/test splits (the same splits for each method). When using node2vec, we tune its p and q parameters as in [3], using grid search with 10-fold cross-validation. We report Accuracy (ACC), AUROC (AUC), and F1 score in Table 3, and discuss results in § 4.6.

1) Trajectory Node Classification is designed to investigate EVO's usefulness for tasks related to trajectories. In this task, we attempt to predict whether each London Tube station services one line or multiple lines. Since trains follow lines, this task is related to trajectories, and requires that a method distinguish between traffic that comes from one line vs. traffic that comes from multiple lines.
Setup. We use our methods with node2vec. Note that 70.9% of stations service one line.
Results. EVO's benefit is clear, as it outperforms all baselines by a statistically significant amount by all metrics, demonstrating that EVO captures trajectories and can identify traffic from multiple lines. In contrast, the baselines only capture *how much* traffic comes through each station. To help explain this, we observe that stations with multiple lines have 6.12 dependencies on average, while stations with only one line have only 2.91. Thus, when a station services more lines, it matters more where a train comes from when determining where it will go next, leading to more dependencies. Indeed, the number of lines station v services is correlated with the size of $\delta(v)$ (0.6381 Pearson correlation coefficient).

2) Geographic Node Classification investigates whether EVO captures geographic information. Unlike the London Tube network, where edges connect nearby stations, flights often directly cross geographic regions (e.g. LAX to JFK), causing the network to not reflect geography. However, itineraries often end in the same region as they began. The task is to predict whether an airport is in the Western or Eastern US.
Setup. We use the Census Bureau's 9 US divisions [1] and consider divisions 1, 2, 3, 5, and 6 (65.7% of airports) as the Eastern region. We use our methods with node2vec.
Results. EVO outperforms all baselines by a statistically significant amount, suggesting that EVO captures geographic information in trajectories, which is lost by networks where edges do not respect geography. Indeed, 84.5% of edges cross regions, but 92.8% of itineraries end in the same region as they started (largely due to round-trip flights).

3) Structural Node Classification seeks to predict whether or not an airport is a hub, as characterized by the FAA [2], which defines hubs in terms of the quantity of enplanements. This is highly related to node degree, and is hence *structural*.
Setup. To capture the structure of nodes, we use xNetMF (we also tried struc2vec, but saw similar results). The baseline accuracy is 62%, as 38% of airports are not hubs.
Results. SNE performs the best, suggesting that degree information is lost in a HON. Indeed, the median size of $\delta(v)$ for airports is 14. However, for the 25 highest degree airports, the median size is 88. Thus, because hubs have more nuanced flying patterns, they also have more dependencies. The hub's degree in G is then distributed across more nodes in H, leading to loss of degree information.

4) Neighborhood Node Classification tests whether dependencies are appropriate for tasks related to node *neighborhoods*. The task is to predict whether a London Tube stop is in an outer or an inner zone. The Tube system is split into 9 zones forming rings around the city center (i.e., zone 1 is the city center, zone 2 is a ring around zone 1, etc.).
Setup. To capture neighborhood information, we use our methods with node2vec. Note that 65.1% of stations are in the inner zone (which we consider to be zones 1, 2, & 3).
Results. Either SNE or SWNE perform the best on this task, likely because Tube trajectories cross 3.2 zones on average, biasing node contexts away from neighborhoods. In contrast, a station is generally in the same zone as its neighbors.

4.5 Clustering

We further investigate RQ1 and RQ2 in an unsupervised setting with more than two classes by clustering nodes and comparing to four ground truth groupings: (1) Tube stations grouped by the

Table 3: For EVO, "*" marks results statistically significantly better than all baselines. For baselines, it marks results stat. significantly better than EVO-max and EVO-avg.
Node Classification (§ 4.4): Task 1: Classify whether a stop services one line or more (trajectories matter). The results suggest that EVO captures high-order dependencies. Task 2: Classify airports by region (geography matters). The results suggest that EVO captures geography via dependencies. Task 3: Classify airports as hubs (structure matters). Results suggest that degree is lost in H. Task 4: Classify tube stops as inner or outer-zone (neighborhoods matter). Results suggest that trajectories can be deceptive.
Clustering (§ 4.5): Results are consistent with classification (except that SWNE outperforms SNE on zones), demonstrating that our observations hold even in an *unsupervised* setting where we consider all categories, unbinarized.

Task	*Metric*	SNE	SWNE	ARWE	EVO-avg	EVO-max
1. Trajectory Node Classification	ACC	0.8259 ± 0.03	0.8481 ± 0.05	0.8506 ± 0.04	0.8321 ± 0.04	0.8901* ± 0.03
	AUC	0.8859 ± 0.04	0.9037 ± 0.04	0.9026 ± 0.04	0.8823 ± 0.05	0.9627* ± 0.02
	F1	0.8792 ± 0.02	0.8960 ± 0.03	0.8969 ± 0.03	0.8886 ± 0.03	0.9254* ± 0.02
2. Geographic Node Classification	ACC	0.6000 ± 0.06	0.5980 ± 0.07	0.6160 ± 0.04	0.7280* ± 0.05	0.7100* ± 0.05
	AUC	0.5233 ± 0.09	0.5951 ± 0.05	0.5179 ± 0.06	0.7791* ± 0.04	0.7611* ± 0.06
	F1	0.2155 ± 0.09	0.2921 ± 0.12	0.1241 ± 0.13	0.4988* ± 0.09	0.4180* ± 0.08
3. Structural Node Classification	ACC	0.8748* ± 0.04	N/A	N/A	0.7730 ± 0.05	0.8107 ± 0.04
	AUC	0.9227* ± 0.04	N/A	N/A	0.8785 ± 0.03	0.9006 ± 0.03
	F1	0.8991* ± 0.03	N/A	N/A	0.8212 ± 0.04	0.8429 ± 0.04
4. Neighborhood Node Classification	ACC	0.9556* ± 0.02	0.9333 ± 0.02	0.9160 ± 0.03	0.9296 ± 0.03	0.9148 ± 0.03
	AUC	0.9916* ± 0.01	0.9867 ± 0.01	0.9791 ± 0.01	0.9770 ± 0.02	0.9684 ± 0.03
	F1	0.9668* ± 0.01	0.9502 ± 0.02	0.9369 ± 0.02	0.9469 ± 0.02	0.9358 ± 0.02
Clustering on Lines	NMI	0.6478 ± 0.01	0.6258 ± 0.02	0.6458 ± 0.01	0.7100* ± 0.01	0.7125* ± 0.02
Clustering on Regions	NMI	0.1441 ± 0.00	0.1245 ± 0.00	0.1096 ± 0.01	0.3856* ± 0.00	0.2880* ± 0.02
Clustering on Hubs	NMI	0.4138* ± 0.00	N/A	N/A	0.0505 ± 0.01	0.2844 ± 0.00
Clustering on Zones	NMI	0.2542 ± 0.02	0.2705* ± 0.02	0.1933 ± 0.03	0.2192 ± 0.01	0.2444 ± 0.02

line they are on, (2) airports grouped by their region, (3) airports grouped by their FAA hub category, and (4) Tube stations grouped by the zone they are in.

Setup. When considering lines as ground truth, we disregard stations which fall on multiple lines, leaving 190 stations. Likewise, when considering zones, we disregard stations in multiple zones, leaving 242. There are 11 Tube lines, 9 zones, 9 airport regions, and 4 airport hub categories (large, medium, small, and non-hub); we use K-Means for clustering with K set to the number of ground truth groups. We use node2vec for all tasks except hubs, where we use struc2vec. For node2vec, since there is no training set, we try the same pairs of p and q as classification, and report results with the best pair. We evaluate the quality of clusterings using Normalized Mutual Information (NMI), and report results in Table 3.

Results. EVO-max and EVO-avg lead to clusters most similar to station lines and airport regions respectively, while either SNE or SWNE lead to clusters most similar to hubs and station zones. The results are consistent with the classification results, suggesting that our observations hold even in the non-binarized, unsupervised clustering setting.

4.6 Discussion

For trajectory related tasks, such as clustering stations by line or classifying an airport's region, EVO significantly outperforms the baselines, demonstrating that EVO captures high-order dependencies. For instance, high-order dependencies capture information about Tube lines, since where passengers go next depends on what line they are on. Furthermore, while edges in an airport network violate geography, high-order dependencies remember the region of origin, since passengers are likely to return there.

However, for many tasks, accurately capturing high-order dependencies can distract from the information that is actually important for that task. We see this in the degradation of performance on structural tasks, where degree information is lost, and likewise on neighborhood tasks, where node contexts are biased *away* from their neighbors.

We found that either element-wise max or average worked best for f in **S3**. We conjecture that when max works best, if a feature has large magnitude for *any* of the embeddings of $\delta(v)$, the property is likely important for v as well, and max preserves this. When average works best, distributional information may be important. This is reminiscent of max- and average-pooling, e.g. [4].

We conjecture that ARWE's poor performance is due to the fact that it aggregates context from variable-order random walks *before* learning. In contrast, EVO builds context for each node in V_H separately and learns from that.

5 CONCLUSION

We propose a modular representation learning framework called EVO, which captures non-Markovian dependencies in node embeddings. We investigate EVO's performance on a wide range of tasks, empirically evaluating when accurately capturing non-Markovian

dependencies is useful for network tasks. We find that if a task is related to trajectories, capturing the non-Markovian dependencies is highly useful, and EVO is an effective way of doing this. However, if a task is unrelated to trajectories, learning over the non-Markovian dependencies can actually be distracting, leading to decreased performance.

ACKNOWLEDGEMENTS

The authors thank Leul Wuletaw Belayneh for his contributions to the project during its early stages, as well as Di Jin and Mark Heimann for their helpful discussions on representation learning. This material is based upon work supported by the National Science Foundation under Grant No. IIS 1845491, an Adobe Digital Experience research faculty award, and an Amazon faculty award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding parties.

REFERENCES

[1] [n. d.]. Census Regions and Divisions of the United States. https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf. ([n. d.]).

[2] 2018. Airport Categories. https://www.faa.gov/airports/planning_capacity/passenger_allcargo_stats/categories/. (2018).

[3] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM.

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[5] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *CIKM*. ACM.

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

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

[8] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. ACM.

[9] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *KDD*. ACM.

[10] Ryan A Rossi, Nesreen K Ahmed, and Eunyee Koh. 2018. Higher-order network representation learning. In *WWW (companion)*.

[11] Martin Rosvall, Alcides V Esquivel, Andrea Lancichinetti, Jevin D West, and Renaud Lambiotte. 2014. Memory in network flows and its effects on spreading dynamics and community detection. *Nature comm* 5 (2014).

[12] Ingo Scholtes. 2017. When is a network a network?: Multi-order graphical model selection in pathways and temporal networks. In *KDD*. ACM.

[13] Jian Xu, Thanuka L Wickramarathne, and Nitesh V Chawla. 2016. Representing higher-order dependencies in networks. *Science advances* 2, 5 (2016).

[14] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE trans. on Big Data* (2018).