Dynamic Embedding-based Methods for Link Prediction in Machine Learning Semantic Network

1st Harlin Lee
Department of Mathematics
University of California, Los Angeles
Los Angeles, USA
harlin@math.ucla.edu

2nd Rishi Sonthalia
Department of Mathematics
University of California, Los Angeles
Los Angeles, USA
rsonthal@math.ucla.edu

3rd Jacob G. Foster
Department of Sociology
University of California, Los Angeles
Los Angeles, USA
foster@soc.ucla.edu

***Abstract*—This paper aims to accelerate scientific discovery by studying link prediction in a semantic network. The nodes are unidentified concepts in machine learning, and the time-stamped edges indicate co-occurrence in scientific papers. Taking advantage of this temporal information, we perform node embedding on the graph at every year from 1994 to 2017, and apply two methods to find features for node pairs: the first method uses a transformer, while the other uses distance metrics combined with known link prediction features. The latter feature extraction technique with a 3-layer multi-layer perceptron achieved an AUC of 0.902 on predicting edges in the 2020 graph. Inspection of the resulting features suggests that the model does indeed pay attention to the dynamic nature of the features, e.g., how node-pair distance in embedding space changes over the years.**

***Index Terms*—science of science, metascience, link prediction, semantic graph, dynamic node embedding**

I. Introduction

The field of artificial intelligence (AI) and machine learning (ML) is exponentially growing with no sign of waning popularity. As the discipline gets larger, it is impossible for an individual researcher to be familiar with the entire body of literature. This forces them to specialize in a sub-field. Such insulation can hinder the birth of ideas that arise from new connections, eventually slowing down scientific progress [1]. As such, discovering fruitful interdisciplinary connections by analyzing scientific publications is an important problem in the science of science, or metascience [2]–[4].

To address this issue, the Science4cast Competition team has released a semantic network dataset as a part of the 2021 IEEE BigData Cup Challenges. In this graph, each of the 64,719 nodes represent a concept or keyword in AI/ML. Two nodes are connected by an undirected edge if the two concepts appear together in a scientific paper. Edges are labeled by the date they were formed (between 1994 and 2017). The goal is to predict which node pairs that are unconnected in 2017 will have an edge by 2020.

We tackle this challenge by focusing on the dynamic nature of the semantic network. Section II sketches how dynamic node embedding is used as a building block to our two approaches to link prediction. Section III discusses the models' performance at the competition and interpretability of the models. Section IV summarizes our findings and points out directions for further study. The notation used throughout this paper is summarized in Table I. We use boldface letters such as $\boldsymbol{x}$ and $\boldsymbol{X}$ to represent vectors and matrices, respectively, and $X[i,j]$ for the (i,j)th element of $\boldsymbol{X}$.

This work was supported by grant TWCF0333 from the Templeton World Charity Foundation.

TABLE I
Notations

Variable	Definition
N	Number of nodes in the semantic graph, i.e. 64719
$\boldsymbol{A}_t$	Adjacency matrix of the semantic graph at year t
$\boldsymbol{A}_t^n$	nth power of $\boldsymbol{A}_t$, or length-n path connections in graph $\boldsymbol{A}_t$
$d_u(\boldsymbol{A}_t)$	Degree of node u in $\boldsymbol{A}_t := \sum_{\ell=1}^{N} A[u,\ell]$
$d_{\max}(\boldsymbol{A}_t)$	maximum node degree of $\boldsymbol{A}_t := \max_{\ell=1,\dots,N} d_\ell(\boldsymbol{A}_t)$
$\boldsymbol{n}_u(\boldsymbol{A}_t)$	128-dimensional embedding vector of node u in $\boldsymbol{A}_t$

II. Methods

In this section, we describe our node feature extraction process, and explain two parallel approaches that use the node embeddings to build a link prediction classifier. Both methods incorporate temporal information to find better features for node pairs, and use multi-layer perceptrons (MLPs) as the final classifier. However, the first approach tries to learn the dynamics of the embedding over the entire time horizon with a transformer, while the second method focuses on the recent past and combines node dynamics with hand-crafted link prediction features. Python code for this project will be available at `https://github.com/HarlinLee/science4cast`.

A. Dynamic Node Embedding

The given dataset contains an unlabeled graph (without any metadata indicating which node in the semantic network corresponds to which scientific concept) and a time stamp for when each edge was created. This means we have access to exactly two pieces of information:

1) the connection or the adjacency matrix of the graph,
2) and the dynamics of this adjacency matrix through time.

With this in mind, we extract various snapshots of the adjacency matrix through time, capturing graphs in the form of $\boldsymbol{A}_t$ for $t = 1994, \dots, 2017$. We then embed each of these

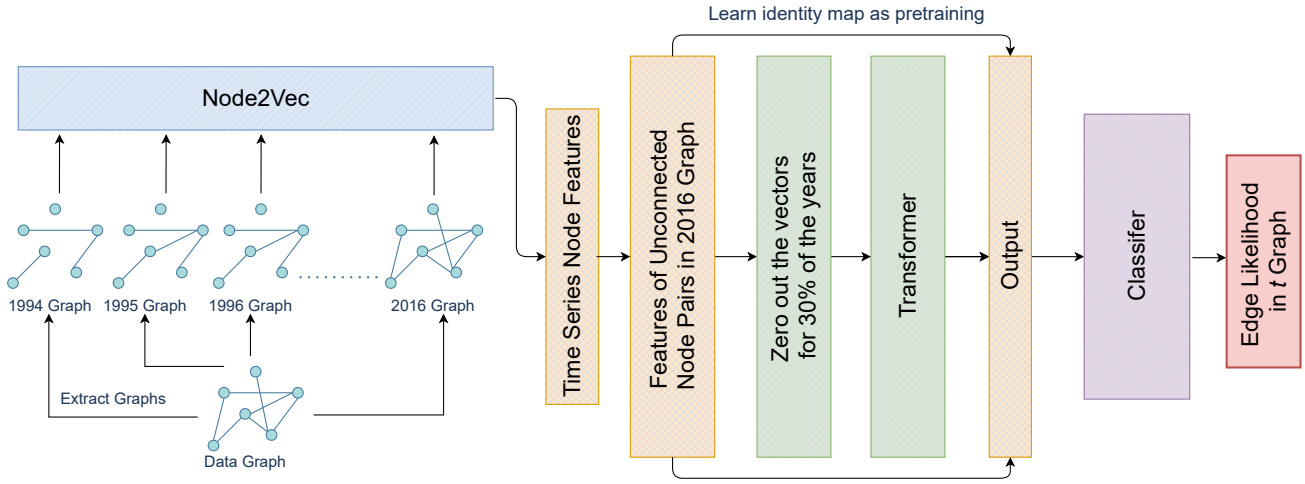

Fig. 1. Outline of method that uses dynamic embeddings and transformers to predict whether two nodes are connected or not. The first step is to extract a graph for each year, and embed them using Node2Vec. The embedding vectors give a time series of features for each node, which we can use to pretrain a transformer. Finally, we classify the pairs of nodes as connected or unconnected with the transformer and a feed-forward network. $t = 2017$ for training and $t = 2020$ for testing.

graphs into 128-dimensional Euclidean space via Node2vec [5]. We use the algorithm as implemented by the StellarGraph Python API [6], and set the biased random walk parameters $p = 2, q = 1$, and walk length $= 5$. Thus, for each node u in the semantic graph, we extract 24 different 128-dimensional vector embeddings $\boldsymbol{n}_u(\boldsymbol{A}_{1994}), \ldots, \boldsymbol{n}_u(\boldsymbol{A}_{2017})$.

B. Approach 1: Transformer

Since we have node embedding vectors from different years, it makes sense to treat them as a time-series signal and try to utilize their dynamics in link prediction. As transformers have performed extremely well in natural language processing tasks [7], we use transformers to learn how the embedding vectors change with time. See Figure 1 for an overview of this approach.

First, since these embedding vectors may not be aligned to each other, we perform a procrustes alignment [8] of the older embeddings $\boldsymbol{n}_u(\boldsymbol{A}_{1994}), \ldots, \boldsymbol{n}_u(\boldsymbol{A}_{2015})$ to the embedding of the 2016 graph $\boldsymbol{n}_u(\boldsymbol{A}_{2016})$. Now we have converted our graphical data into time series data, where each node has 23 snapshots of the 128-dimensional embedding vector through time. Note that we have set aside the 2017 graph for now.

We then train a transformer to help classify node pairs. For a given unconnected node pair in 2016, the input to the transformer is a pair of embedding sequences. Given this input, we then mask out 30% of the vectors in this sequence. That is, we pick 30% of the years for the first node and 30% of the years for the second node (independently) and set all elements of those vectors to 0. We then pretrain the transformer to reconstruct those inputs, with the hope that this would help learn the dynamics of the vectors. For the transformer, the encoder and decoder had 3 layers each; we used 128 as the embedding dimension, 2048 as the feedforward dimension and 8-headed attention.

Once we pretrain our transformer, we add a 3-layer ReLU network with hidden layers of size $(128 \times 128, 128 \times 2, 2 \times 2)$ as a classifier on top and then fine tune the model to predict whether the pair of nodes are connected or not in 2017. Finally, at the testing phase, the fixed model takes in features of unconnected node pairs in 2017 and predicts their edge likelihoods in 2020.

C. Approach 2: Embedding Distances and Hand-crafted Network Features

In this approach, we calculate pairwise distances between the embedding vectors to reduce feature dimensionality. We then add some well-established, hand-crafted link prediction features to increase the prediction accuracy. This pipeline is visualized in Figure 2.

For a given year, we derive two embedding features and 15 network features for each node pair, as described in Table II. If the divider is zero, e.g. $\min(d_i, d_j)$ in Hub Promoted Index, then the feature is set to 0 for the node pair. The embedding features are calculated from the past 5 years, i.e. $\boldsymbol{A}_{t-4}, \ldots, \boldsymbol{A}_t$, and the 15 hand-crafted network features are derived from the past 3 years, $\boldsymbol{A}_{t-2}, \ldots, \boldsymbol{A}_t$, yielding a total of $2 \times 5 + 15 \times 3 = 55$ features for each node pair. Note that this feature extraction process needs to be done twice, once for the training set with $t = 2014$, and once for the test set with $t = 2017$. Once the features are derived, a simple 3-layer MLP is fitted to the training set to predict edge likelihood in the $t + 3$ graph.

III. Results and Discussions

We first discuss the performances of both approaches in the 2020 graph, then analyze the features and models of the second approach in more detail.

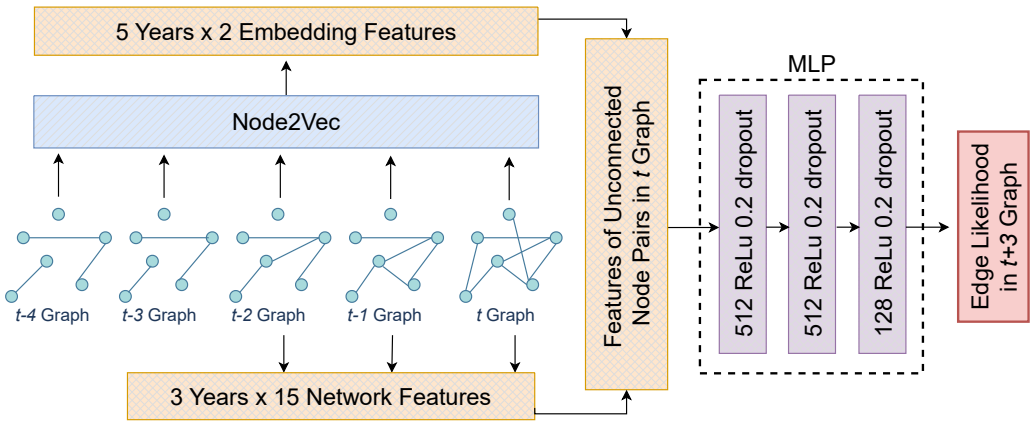

Fig. 2. Outline of method that combines node-embedding distances and hand-crafted link prediction features. $t = 2014$ for training and $t = 2017$ for testing. The embedding features and network features are described in Table II.

TABLE II
TWO EMBEDDING AND 15 NETWORK FEATURES FOR A GIVEN YEAR t AND NODE PAIR (i, j). $\boldsymbol{n} := \boldsymbol{n}(\boldsymbol{A}_t), d := d(\boldsymbol{A}_t)$ UNLESS OTHERWISE STATED.

Feature	Description	
ℓ_2 norm difference	$\|\boldsymbol{n}_i - \boldsymbol{n}_j\|_2^2/128$	
Cosine similarity	$\langle \boldsymbol{n}_i, \boldsymbol{n}_j \rangle / \|\boldsymbol{n}_i\|_2 \|\boldsymbol{n}_j\|_2$	
Normalized node degree i	$d_i/d_{\max}$	
Normalized node degree j	$d_j/d_{\max}$	
Preferential Attachment	$d_i d_j / d_{\max}^2$	[9]
Node i Katz centrality, Katz_i	$\alpha \sum_{\ell=1}^{N} A_t[i,\ell]\mathsf{Katz}_\ell + 1$	[10]
Node j Katz centrality, Katz_j	$\alpha \sum_{\ell=1}^{N} A_t[j,\ell]\mathsf{Katz}_\ell + 1$	[11]
Common Neighbors	$A_t^2[i,j] / \max_{m,\ell}\left(A_t^2[m,\ell]\right)$	
Leicht-Holme-Newman Index	$A_t^2[i,j]/(d_i d_j)$	[12]
Salton Cosine Formula	$A_t^2[i,j]/\sqrt{d_i d_j}$	[13]
Hub Promoted Index	$A_t^2[i,j]/\min(d_i, d_j)$	[14]
Hub Depressed Index	$A_t^2[i,j]/\max(d_i, d_j)$	[15]
Sørensen Index	$A_t^2[i,j]/(d_i + d_j)$	[16]
Jaccard Coefficient	$A_t^2[i,j]/(d_i + d_j - A_t^2[i,j])$	[17]
Number of length-2 paths to i	$d_i(\boldsymbol{A}_t^2)/d_{\max}(\boldsymbol{A}_t^2)$	
Number of length-2 paths to j	$d_j(\boldsymbol{A}_t^2)/d_{\max}(\boldsymbol{A}_t^2)$	
Preferential Attachment for paths	$d_i(\boldsymbol{A}_t^2) d_j(\boldsymbol{A}_t^2)/d_{\max}(\boldsymbol{A}_t^2)^2$	

A. Link Prediction Performance

In the first approach, with a dynamic graph and transformers, we noticed that the classifier has a cross entropy training and validation loss of roughly 0.250. That is, the validation loss was similar to the training loss, and we were likely not over fitting. However, in training, we used information from 1994 up to (and including) 2016 to predict whether edges were connected in the 2017 graph, while we predicted edges in the 2020 graph using the same information for our test set. Hence there is a distributional mismatch between the two. We postulate that it is due to this mismatch that the test AUC for this method was only 0.73. Additionally, the alignment of the vectors using procrustes, may have destroyed some of the dynamical information and that may be another reason for the low performance.

The second approach, with embedding distances and hand-crafted features, achieved a test AUC of 0.902. Preliminary results showed that this value is not improved by changing the number of years used in feature extraction (5 and 3) or the MLP architecture, but more systematic experiments are needed before concluding that the AUC cannot be improved by tuning hyperparameters. This AUC is also higher than the values attained by the embedding features only or the hand-crafted features only. This suggests that there is likely complementary information between them.

B. Understanding Dynamic Features and Model Behavior

The non-convexity and hierarchical structure of the MLP make it difficult to analyze its interaction with the features as one might with a linear regression model. Therefore, we inspect a few of the 17 features in Table II and observe their trends across different groups of node pairs to better understand the dynamic features and the model behavior.

In order to construct these groups, we trained our MLP (see Figure 2) on 95% of the training set, and applied it to the remaining 5%, i.e. the validation set, to get the *model's predicted edge likelihood* p. Then, we divided the validation set into three groups:

- $p < 0.1$: low group has 45,408 node pairs.
- $0.1 \leq p \leq 0.85$: mid group has 173,529 node pairs.
- $0.85 \leq p$: high group has 69,008 node pairs.

Figure 3 visualizes select feature values across time for each group. The columns correspond to the three groups, and rows to features. The horizontal axes are years, the vertical axes are feature values, the solid line is the mean value, and the opaque areas are one standard deviation away from the mean.

The first two rows show the embedding features over 5 years. Recall that identical vectors will have minimum ℓ_2 distance but maximum cosine similarity. We observe a clear downward trend in embedding ℓ_2 distance for mid and high groups on the first row, and upward trends in cosine similarity

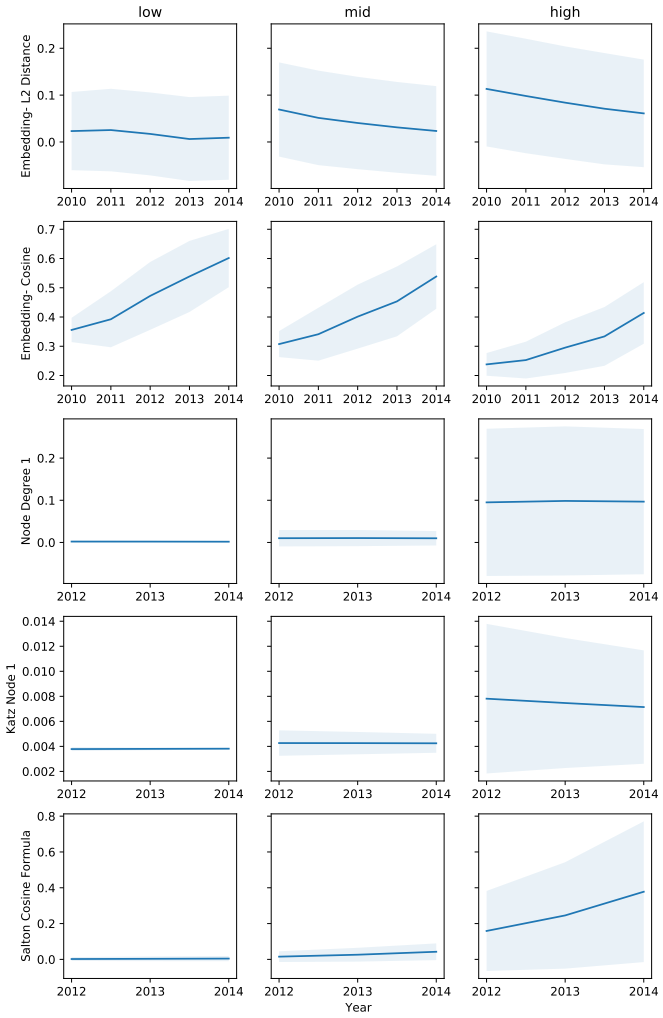

Fig. 3. A subset of validation set features from Table II plotted over time, grouped by the model's predicted edge likelihood. The model in Figure 2 agrees that nodes that are getting closer in the embedding space over time are more likely to form an edge in the future. The solid line is the mean value, and the opaque areas are one standard deviation away from the mean.

for all groups. These trends match our intuition that nodes that are getting closer in the embedding space over time are more likely to form an edge in the future. Somewhat surprisingly, the model seems to assign higher probability to pairs that "start off worse", i.e., those with higher ℓ_2 distance or lower cosine similarity. One possible reason is that nodes that start off close by are ideas that people have already tried to connect and failed to do; note that there is a sample bias as we are only looking at node pairs that have *not* been connected by year t. This persistent failure might be due to conceptual difficulty of the connection, or to conceptual incompatibility (i.e., the ideas are too similar to be profitably linked). Nodes that started off distant, by contrast, correspond to ideas that people have not tried to connect yet. As the distance decreases, people notice that the concepts could be connected, attempt to connect them, and succeed.

The third and fourth rows show normalized node degree and Katz centrality for one of the two nodes; results are similar for either node in a given pair. In both cases, there are no clear trends over time, but the mean in high group is slightly higher than that of others, and the variance is much larger. This could be a reflection of the model recognizing the "interesting nodes": non-interesting nodes are not interesting in the same way, e.g. isolated nodes, and they are not likely to make edges. The Salton's cosine formula of the high group appear to increase over time; it also differs from others in terms of its high variance and mean away from zero.

IV. Conclusions

We used dynamic embedding to extract time-varying features for each node in a semantic network, and demonstrated that adding distance-related summary statistics of the embedding vectors can improve upon the predictive power of existing link prediction features. Preliminary interpretability analysis showed that the MLP takes the dynamic nature of the node pair features into account when predicting edges, which aligns with our motivation behind using dynamic embedding-based methods. In the future, a more rigorous experiment could help determine which of the features are essential, e.g. leaving one feature out at a time; how far back we should look into the past, e.g. 3 years vs. 5 year; and what other embedding distance metrics are useful. We suspect that increasing the granularity of the snapshots— for example, embedding graphs for every month instead of every year— will increase the performance of our approaches, but that requires more computing resources.

Acknowledgment

The authors thank the Science4cast Competition organizers for creating and sharing this dataset.

References

[1] A. Rzhetsky, J. G. Foster, I. T. Foster, and J. A. Evans, "Choosing experiments to accelerate collective discovery," *Proceedings of the National Academy of Sciences*, vol. 112, no. 47, pp. 14 569–14 574, 2015.

[2] S. Fortunato, C. T. Bergstrom, K. Börner, J. A. Evans, D. Helbing, S. Milojević, A. M. Petersen, F. Radicchi, R. Sinatra, B. Uzzi, A. Vespignani, L. Waltman, D. Wang, and A.-L. Barabási, "Science of science," *Science*, vol. 359, no. 6379, p. eaao0185, Mar. 2018. [Online]. Available: https://www.sciencemag.org/lookup/doi/10.1126/science.aao0185

[3] J. Evans and J. Foster, "Metaknowledge," *Science (New York, N.Y.)*, vol. 331, pp. 721–5, 02 2011.

[4] M. Krenn and A. Zeilinger, "Predicting research trends with semantic and neural networks with an application in quantum physics," *Proceedings of the National Academy of Sciences*, vol. 117, no. 4, pp. 1910–1916, 2020.

[5] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[6] C. Data61, "Stellargraph machine learning library," https://github.com/stellargraph/stellargraph, 2018.

[7] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[8] J. C. Gower, "Generalized procrustes analysis," *Psychometrika*, vol. 40, pp. 33–51, 1975.

[9] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.
[10] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.
[11] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 2008, pp. 11–15.
[12] E. A. Leicht, P. Holme, and M. E. Newman, "Vertex similarity in networks," *Physical Review E*, vol. 73, no. 2, p. 026120, 2006.
[13] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. mcgraw-hill, 1983.
[14] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, "Hierarchical organization of modularity in metabolic networks," *science*, vol. 297, no. 5586, pp. 1551–1555, 2002.
[15] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, no. 4, pp. 623–630, 2009.
[16] T. A. Sorensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons," *Biol. Skar.*, vol. 5, pp. 1–34, 1948.
[17] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

APPENDIX

Figure 4 is the full version of Figure 3. Figure 5 uses the same validation set as Figure 4, but divides the groups based on their true labels instead of model prediction. Figure 6 repeats the process in Figure 4 but for the test set, which we don't have the true link prediction labels for.

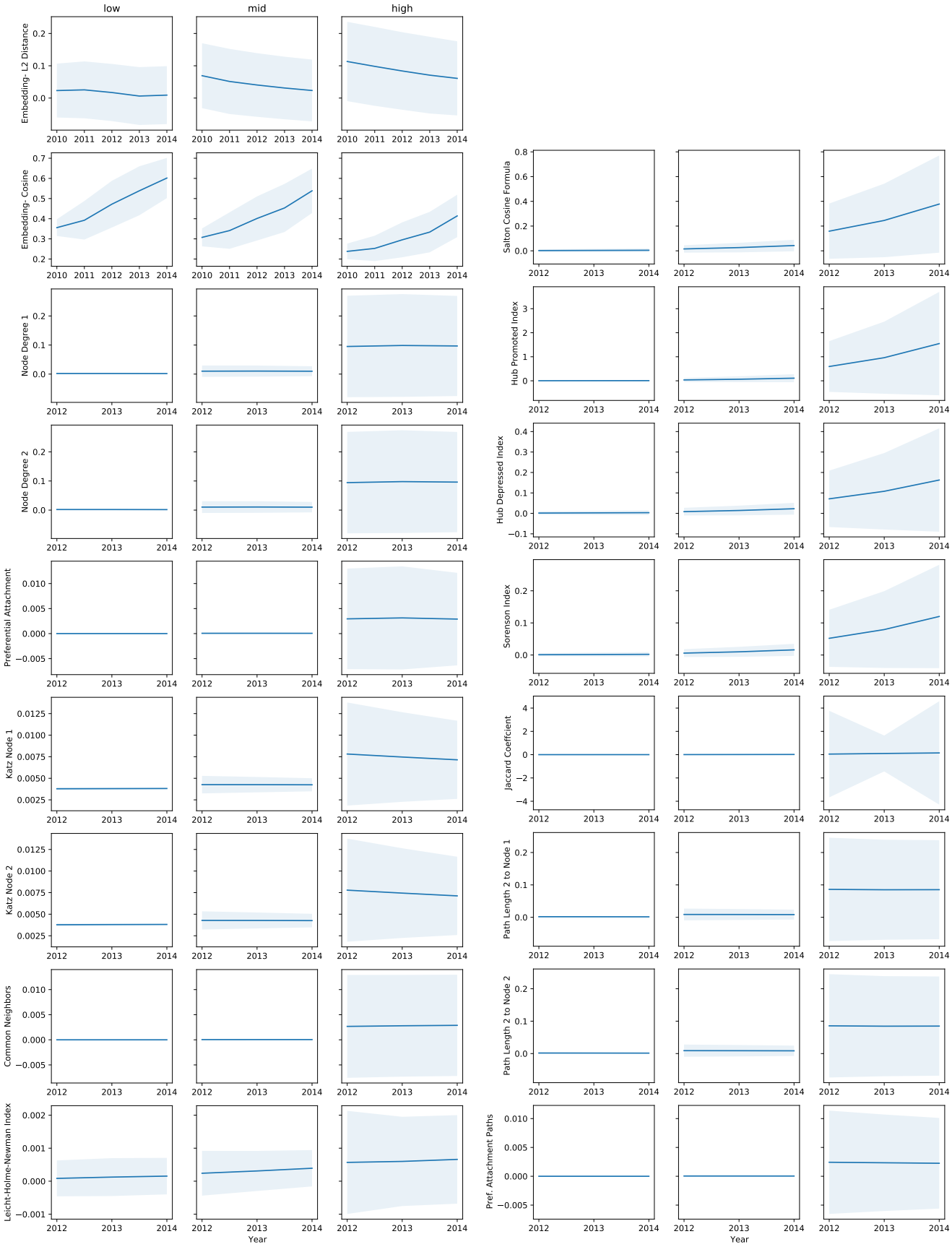

Fig. 4. Feature over time on validation set node pairs.

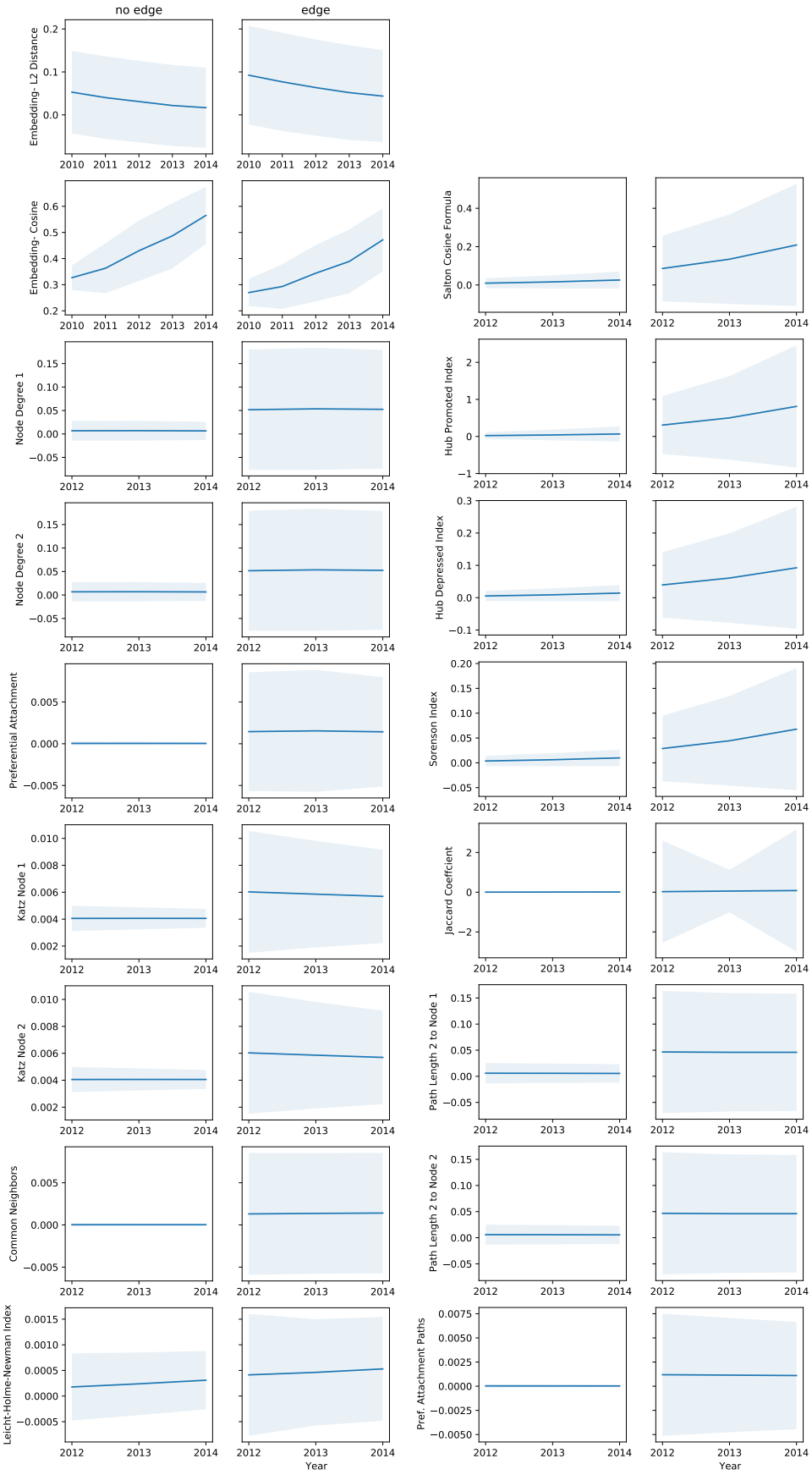

Fig. 5. Feature over time on validation set node pairs, but divided by the true label.

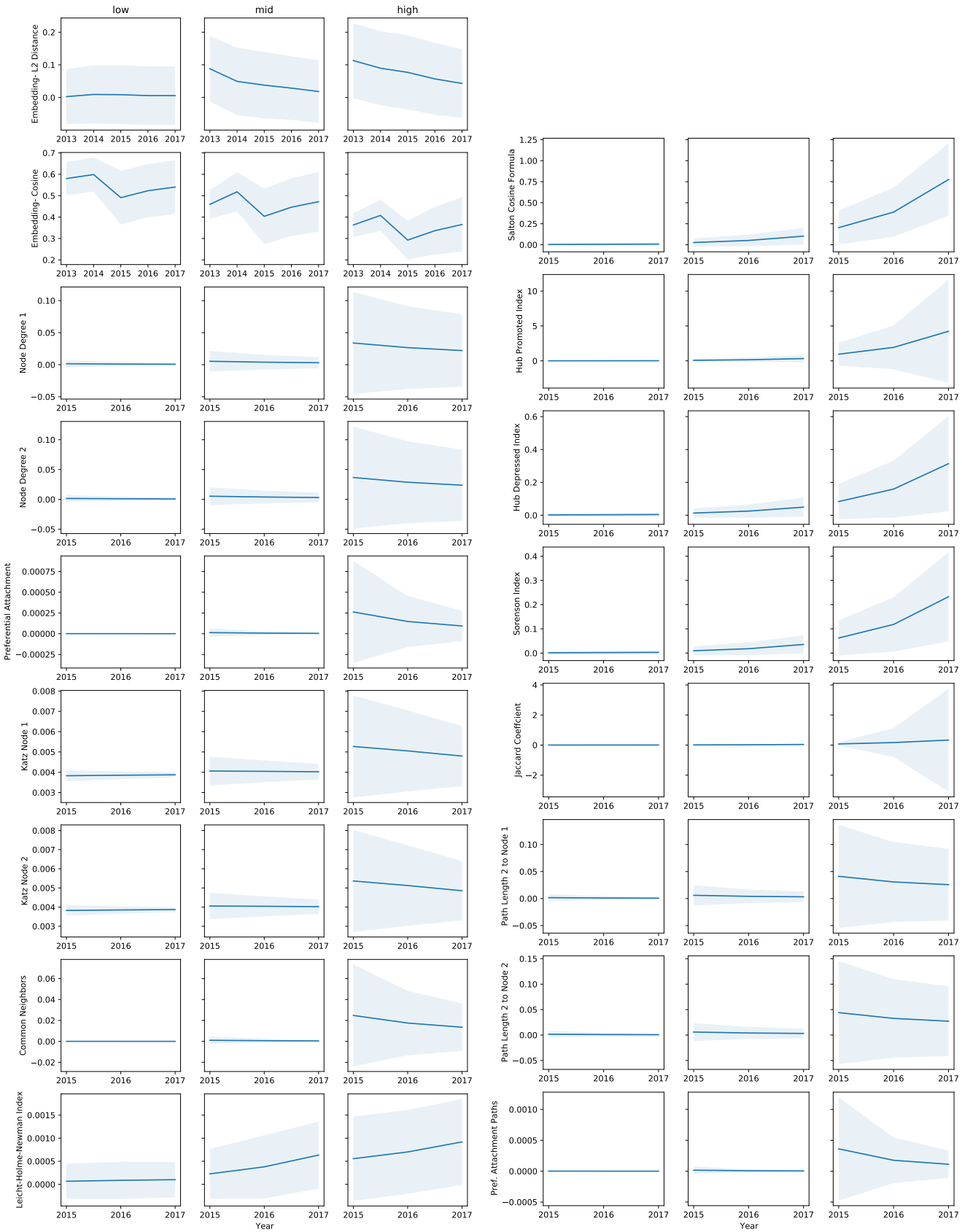

Fig. 6. Feature over time on test set node pairs.