KU LEUVEN

Combining Temporal Aspects of Dynamic Networks with node2vec for a more Efficient Dynamic Link Prediction

Sam De Winter, Tim Decuypere, Sandra Mitrović, Bart Baesens and Jochen De Weerdt

Department of Decision Sciences and Information Management, KU Leuven

28 August 2018, DyNo Workshop ASONAM

Barcelona, Spain

Outline

1. Introduction
2. Relevant literature
3. Research questions
4. Methodology
5. Results
6. Conclusion

Introduction

Link Prediction:

- *“Given a snapshot of a social network G(V,E) at time t, we seek to accurately predict the edges that will be added to the network during the interval from time t to a given future time t*.”*

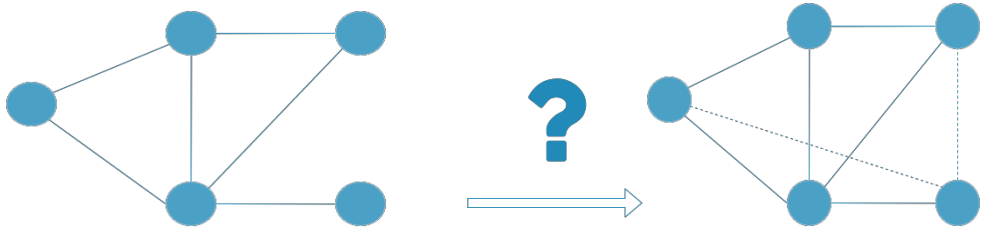
  - Liben-Nowell and Kleinberg (2007)

Applications

- Recommender systems
  - Facebook suggested friends
  - Amazon product bundling
- Protein interactions in bioinformatics, …

Static → Dynamic

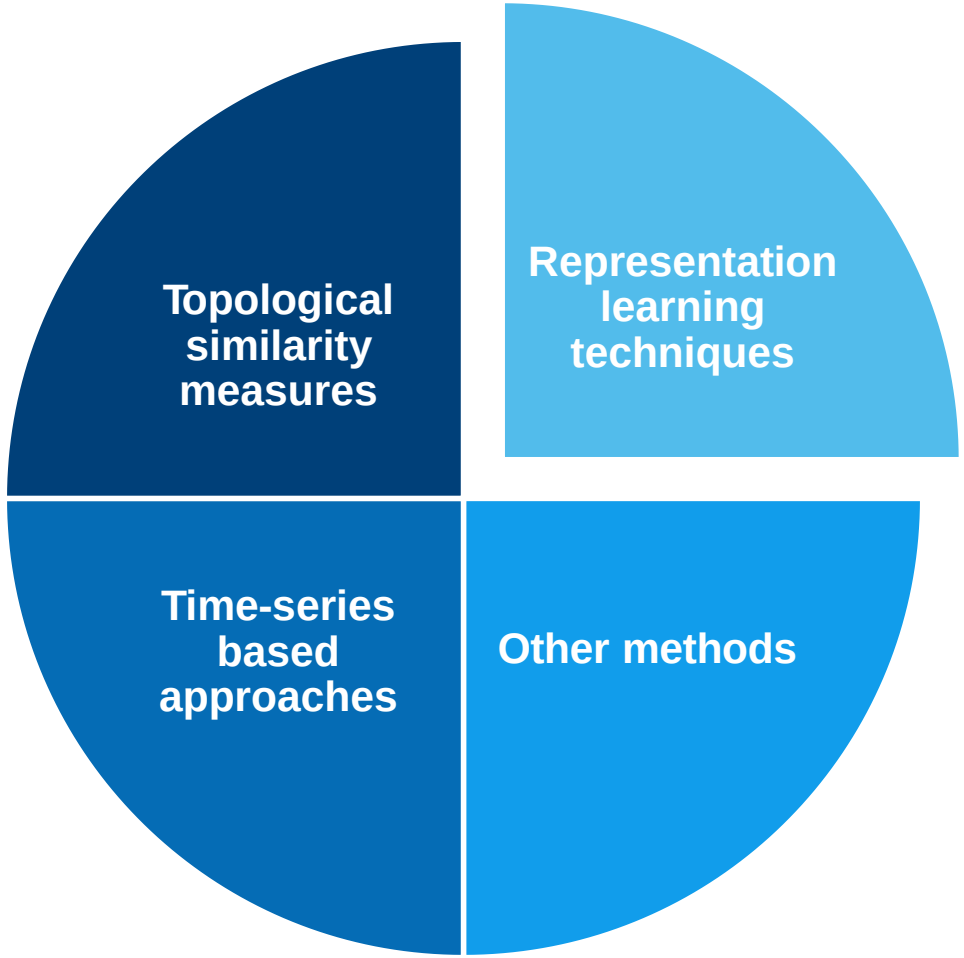
- Many studies consider only static graphs
  - Only study a graph one point in time
  - No capturing of temporal effects
  - Sometimes only predict link(s) within a snapshot
  - Assume that V stays the same (and only E changes)



Real-world scenario

- Network structure (both V and E) and node attributes are all dynamically evolving and at an unseen rate

Relevant literature

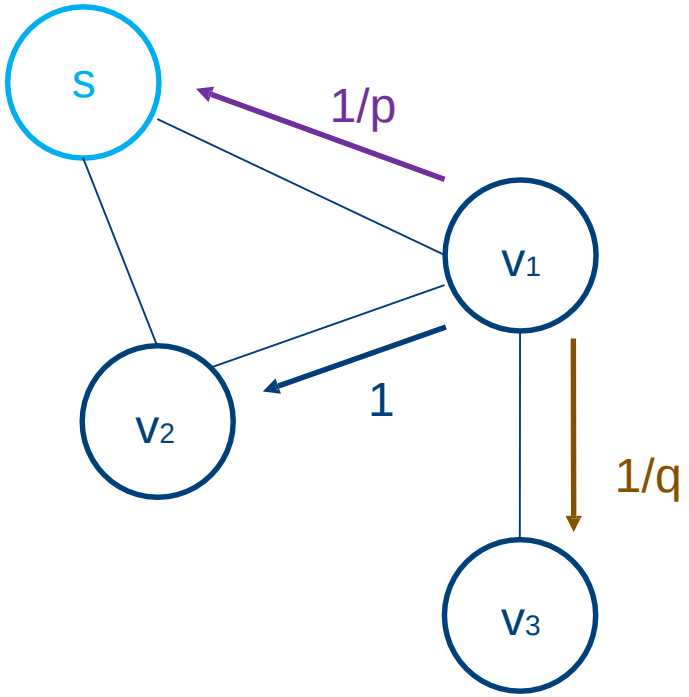


Representation learning: node2vec

Grover & Leskovec (2016)

- **Goal:** Embed nodes such that likelihood of preserving network neighborhoods of the nodes is maximized
- Capture neighborhood by random walks
- Random walks are biased by parameters *p* and *q*

Parameters p & q



- Starting node *s*
- Random walk: first walk to v_1
- Which node to visit next?
  - Node *v2* with weight *1*
  - Back to *s* with weight *1/p*
  - Node *v3* with weight *1/q*

  (based on shortest path length between candidate node and node *s*)
- Low *p* value: Breadth First Sampling
- Low *q* value: Depth First Sampling

Representation learning: Continuous-time dynamic network embeddings

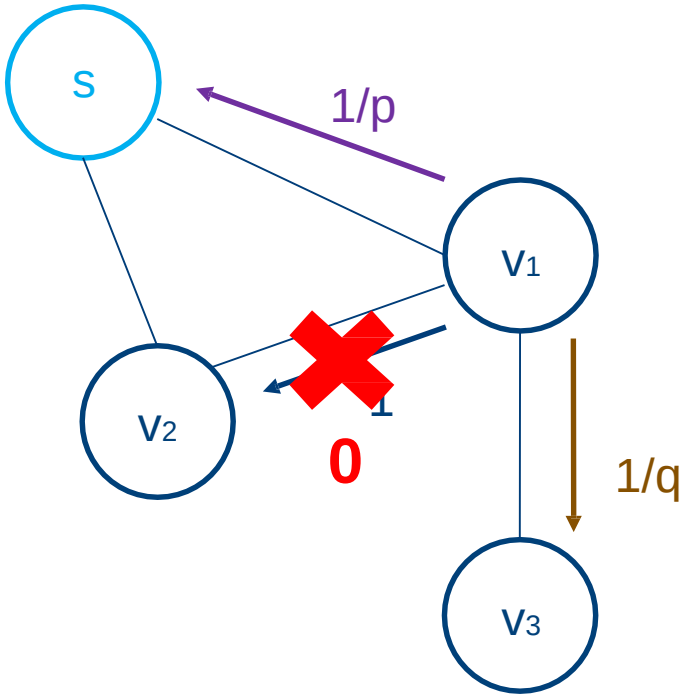
Nguyen et al. (2018)

Goal: featurize nodes in the network better by including the time dimension

- Node2vec with temporally valid random walks
- Temporally valid random walk = the random walk has to travers edges that have timestamps that are increasing in time

Temporally valid random walks

Interaction $v_1 - v_2$ happened earlier than $s - v_1$
=> Probability of going from v_1 to v_2 is 0



Interactions between nodes		
Source	Destination	Timestamp
v1	v2	1
s	v1	2
s	v2	3
v1	v3	4
v1	s	5
v2	s	6
v3	v1	7

Research questions

RQ1: *Given the results of a static approach using node2vec, can a dynamic implementation improve results?*

Two dynamic implementations considered:

- Snapshot - based
- Temporal random walks - based

RQ2: *What are the effects of the parameters of node2vec and how does this differ among networks?*

Methodology

Six steps to conduct the research in a dynamic network environment

1st step: Dividing the dataset in snapshots

2nd step: Graph creation and preparation

3rd step: Calculating the feature vector for each snapshot

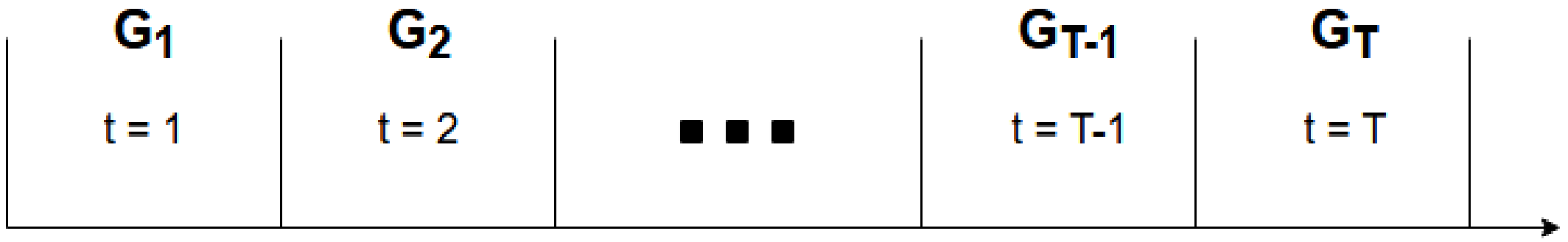
4th step: Node feature vectors and combing the snapshots

5th step: Performing the link prediction (supervised)

6st step: Assessing the performance of the algorithm

1st step: Dividing the dataset in snapshots

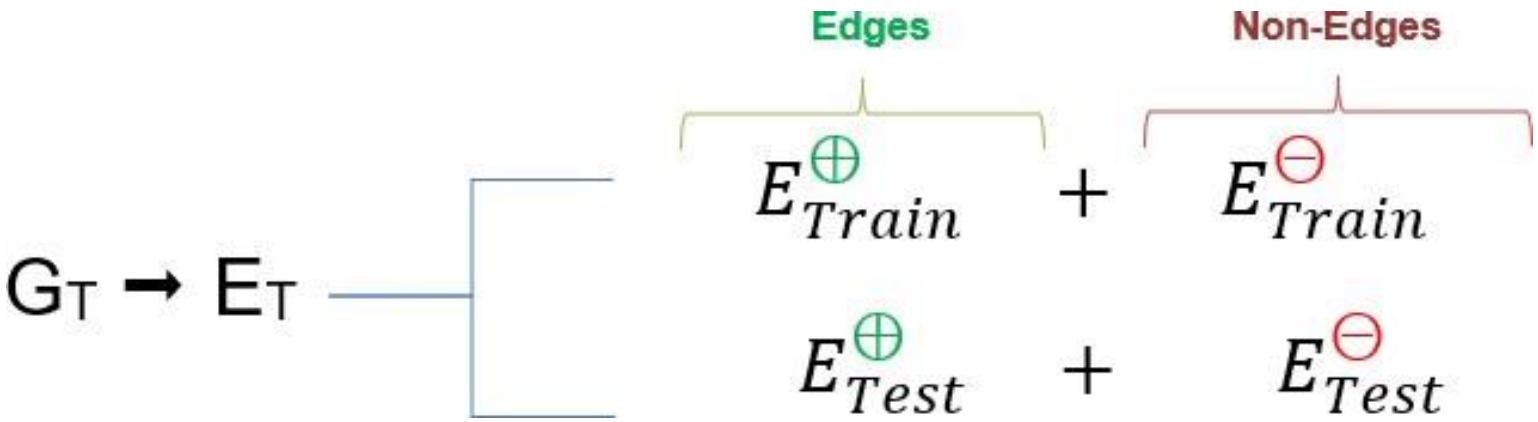
- We divide the dataset in T snapshots ($t \in [1, T]$)
- For each snapshot a graph $G_t(V,E)$ is created
- We use snapshots [1,T-1] as training data
- We will predict links in snapshot T (prediction dataset)
- Static approach: [1,T-1] is seen as one (big) snapshot



2nd step: Graph creation and preparation

Modify the graphs in three ways:

- Define a 70/30 training-test split
- Sample an equal number of non-edges (obtain a balanced prediction set)
- Remove test edges from graphs G_t with $t \in [1, T]$



KU LEUVEN

3rd step: Calculating the feature vector for each snapshot

<u>All aproaches</u>

- Create feature vectors for all snapshots using node2vec

<u>Continuous method</u>

- Temporal random walks
- Random walks that can only traverse nodes with a higher timestamp than the one that was performed before
- More formally, random walk with sequence of edges $\{(V_{i1}, V_{i2}, t_{i1}), (V_{i2}, V_{i3}, t_{i2}), \ldots, (V_{iL-1}, V_{iL}, t_{iL-1})\}$ is temporal random walk if:

  $t_{i1} \leq t_{i2} \leq \ldots \leq t_{iL-1}$

4th step: Node feature vectors and combing the snapshots

- Up till now: feature vectors of individual nodes in different snapshots
- Use Hadamar operator to form edge feature vectors
- Simple concatenation to combine feature vectors from different snapshots

$$E^{\oplus\ominus}_{Train} = \begin{bmatrix} EF^{\oplus}_{Tr\_1} & EF^{\oplus}_{Tr\_2} & EF^{\oplus}_{Tr\_3} & \cdots & EF^{\oplus}_{Tr\_(T-1)} & 1 \\ EF^{\ominus}_{Tr\_1} & EF^{\ominus}_{Tr\_2} & EF^{\ominus}_{Tr\_3} & \cdots & EF^{\ominus}_{Tr\_(T-1)} & 0 \end{bmatrix}$$

$$E^{\oplus\ominus}_{Test} = \begin{bmatrix} EF^{\oplus}_{Test\_1} & EF^{\oplus}_{Test\_2} & EF^{\oplus}_{Test\_3} & \cdots & EF^{\oplus}_{Test\_(T-1)} & 1 \\ EF^{\ominus}_{Test\_1} & EF^{\ominus}_{Test\_2} & EF^{\ominus}_{Test\_3} & \cdots & EF^{\ominus}_{Test\_(T-1)} & 0 \end{bmatrix}$$

5th step: Performing the link prediction (supervised)

- Three classifiers
- Addional two baseline measures used in the literature:
  - Common neighbors
  - Adamic Adar

Logistic Regression		**Random Forests**		**Gradient Boosting**	
Penalty	Ridge reg. (L2)	Criterion	Gini	Criterion	MSE
Solver	Liblinear	N trees	10	Learning rate	0.50
Tolerance	1e-4	Max feat.	Auto	Loss func.	Deviance
Max iter.	100	Max depth	None	Max depth	8

6st step: Assessing the performance of the algorithm

Two evalution methods:

- Area-Under Curve (AUC) → Receiver operating characteristic (ROC)
- Average Precision (AP) → Precision-Recall curve

Experimental evaluation

Sparse networks

Datasets:

Enron employees	Radoslaw email network	Facebook forum interactions	Dublin contacts interaction network	Reality call mobile network
• 2 years • 8 snapshots • 150 nodes • 1588 edges • Avg. Degree = 21.17 • Density = 0.14210	• 9 months • 9 snapshots • 167 nodes • 3251 edges • Avg. Degree = 38.93 • Density = 0.23454	• 6 months • 6 snapshots • 899 nodes • 7046 edges • Avg. Degree = 15.68 • Density = 0.01745	• 8 weeks • 8 snapshots • 6454 nodes • 24097 edges • Avg. Degree = 7.47 • Density = 0.00116	• 14 weeks • 7 snapshots • 6416 nodes • 7250 edges • Avg. Degree = 2.26 • Density = 0.00035

Dense networks

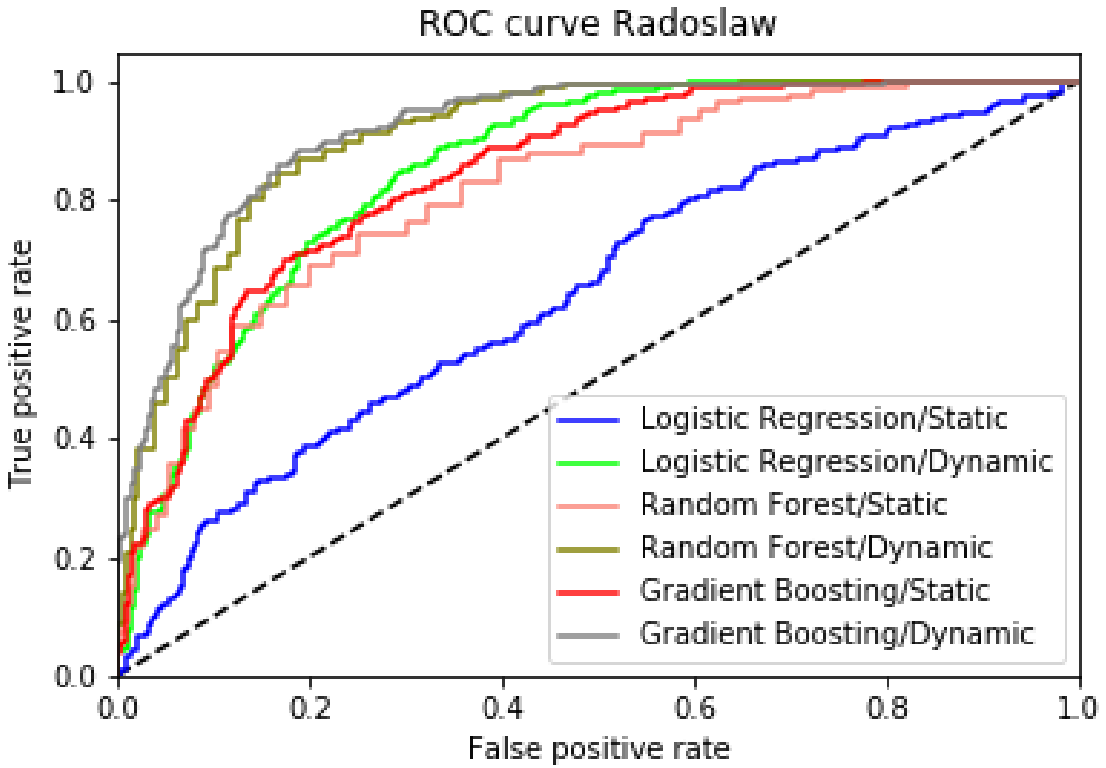
Dense networks

		AUC Scores			Average Precision		
		Static	Dyn-Snap	Dyn-Cont	Static	Dyn-Snap	Dyn-Cont
Enron Employees	Adamic Adar	0.89700	-	-	0.89608	-	-
	Jaccard Coef.	0.87518	-	-	0.86956	-	-
	Logistic reg.	0.81188	0.87078	0.83418	0.80327	0.87468	0.82919
	Random forest	0.88919	0.91122	0.89044	0.88789	0.91418	0.89208
	Grad. boosting	0.88331	0.90295	0.88465	0.88100	0.90472	0.88428
Radoslaw	Adamic Adar	0.90281	-	-	0.89264	-	-
	Jaccard Coef.	0.83071	-	-	0.79369	-	-
	Logistic reg.	0.59344	0.85408	0.67693	0.58303	0.81708	0.66660
	Random forest	0.84522	0.91465	0.85185	0.82271	0.89259	0.82632
	Grad. boosting	0.85897	0.92066	0.85897	0.84176	0.90877	0.84084
Facebook Forum	Adamic Adar	0.65336	-	-	0.64871	-	-
	Jaccard Coef.	0.63214	-	-	0.56346	-	-
	Logistic reg.	0.85467	0.95684	0.92299	0.86147	0.95553	0.93286
	Random forest	0.83152	0.94639	0.91621	0.81914	0.93952	0.91379
	Grad. boosting	0.80719	0.92884	0.88821	0.80932	0.91886	0.88873

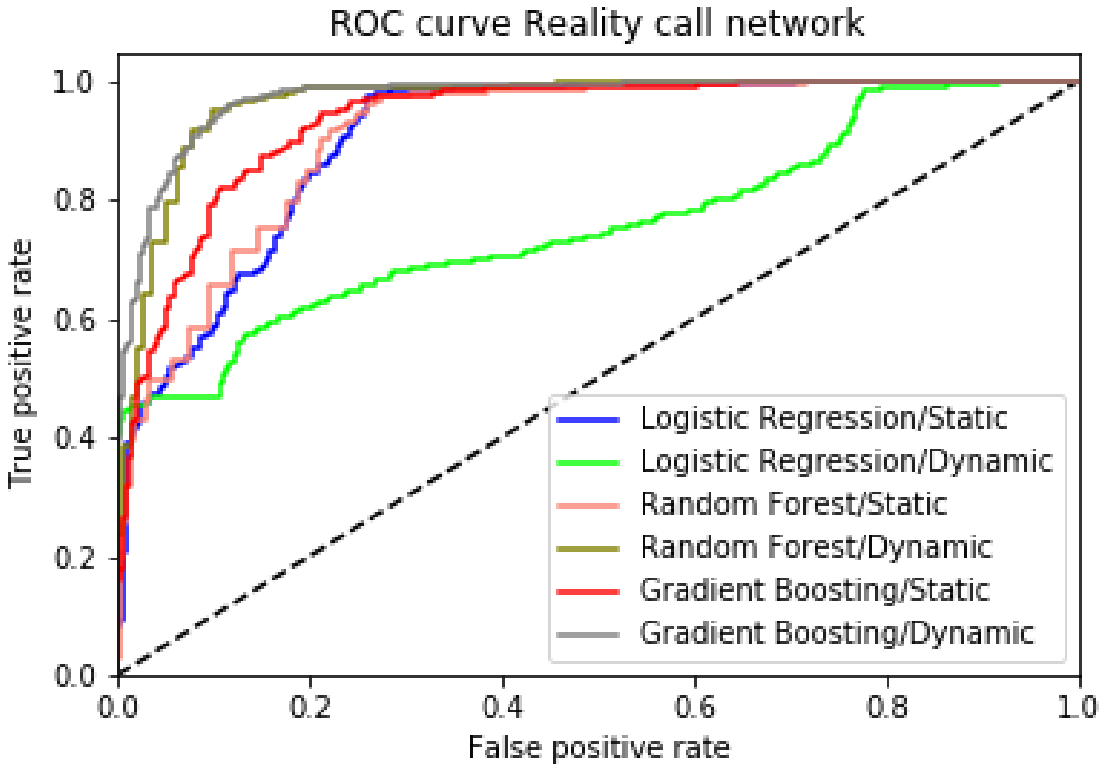
Sparse networks

		AUC Scores			Average Precision		
		Static	Dyn-Snap	Dyn-Cont	Static	Dyn-Snap	Dyn-Cont
Contacts Dublin	Adamic Adar	0.91421	-	-	0.91421	-	-
	Jaccard Coef.	0.91414	-	-	0.91404	-	-
	Logistic reg.	0.85360	0.54903	0.85801	0.70126	0.47283	0.70706
	Random forest	0.98714	0.98687	0.98664	0.97509	0.97451	0.97379
	Grad. boosting	0.98038	0.98611	0.98096	0.97291	0.97490	0.97208
Reality Call	Adamic Adar	0.51923	-	-	0.52103	-	-
	Jaccard Coef.	0.51897	-	-	0.50884	-	-
	Logistic reg.	0.90432	0.75741	0.91105	0.89345	0.81588	0.90620
	Random forest	0.91992	0.97114	0.92288	0.90424	0.96546	0.90463
	Grad. boosting	0.91867	0.97426	0.92841	0.90611	0.97113	0.91970

ROC-curves of a dense and a sparse network



Dense network



Sparse network

KU LEUVEN

Discussion

- In general:
  - Dynamic Snaphshot node2vec > Dynamic Continuous node2vec > Static node2vec
  - Gradient boosting and random forests are better than logistic regression

➔ Time dimension does contain information not captured in a static approach

p & q parameters: dense network (Radoslaw)

	P = 0.25	P = 0.5	P = 0.75	P = 1
Q = 0.1	0.91063	0.91512	0.90678	0.89603
Q = 0.5	0.91149	0.91904	0.92239	0.91256
Q = 1	0.92471	0.92019	0.92176	0.92544
Q = 2	0.92992	0.92457	0.93046	0.92669
Q = 5	0.91859	0.91795	0.92725	0.93228
Q = 10	0.92473	0.92242	0.92643	0.92812
Q = 100	0.92712	0.92359	0.92936	0.92741

Higher q parameter is better ➔ Breadth First Sampling

p & q parameters: sparse network (Reality call)

	P = 0.25	P = 0.5	P = 0.75	P = 1
Q = 0.1	0.97881	0.97092	0.97649	0.97501
Q = 0.5	0.96826	0.97468	0.96076	0.97875
Q = 1	0.97406	0.97135	0.97064	0.97357
Q = 2	0.97521	0.96517	0.97153	0.96462
Q = 5	0.96488	0.96617	0.97255	0.9717
Q = 10	0.96692	0.97181	0.96551	0.96542
Q = 100	0.96625	0.96935	0.96894	0.96685

Lower q parameter is better ➔ Depth First Sampling

Conclusion

- We propose two different strategies of extending a well-known node2vec approach from purely static to dynamic link prediction:
  - Snapshot approach
  - Continuous approach with temporal random walks

- Findings:
  - Taking into account dynamic aspect improves results
  - Snapshot approach performs better than the temporal walks one
  - Gradient boosting / Random Forests preferred to Logistic Regression
  - In smaller more dense networks → Higher in-out param. q
  - In larger more sparse networks → Lower in-out param. q
  - The Return-parameter p has a less significant influence on results

Further Research

- Combine Dynamic & Continuous into one method
- Optimise and test method on large networks for practical applications

References

1. Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855-864). ACM.
2. Güneş, İ., Gündüz-Öğüdücü, Ş., & Çataltepe, Z. (2015). Link prediction using time series of neighborhood-based node similarity scores. Data Mining and Knowledge Discovery, 30(1), 147–180.
3. Liben-Nowell, D., & Kleinberg, J. (2007). The link prediction problem for social networks. Proceedings of the Twelfth International Conference on Information and Knowledge Management - CIKM ’03, 556–559.
4. Lichtenwalter, R. N., Lussier, J. T., & Chawla, N. V. (2010). New perspectives and methods in link prediction. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’10, 243.
5. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
6. Moradabadi, B., & Meybodi, M. R. (2017). A novel time series link prediction method: Learning automata approach. Physica A: Statistical Mechanics and Its Applications, 482, 422–432.
7. Nguyen, C. H., & Mamitsuka, H. (2012). Latent feature kernels for link prediction on sparse graphs. IEEE transactions on neural networks and learning systems, 23(11), 1793-1804.
8. Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Kim, S., & Koh, E. (2018). Continuous-Time Dynamic Network Embeddings. Proceedings of the WWW ’18 Companion.
9. Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710). ACM.
10. Rahman, M., & Hasan, M. Al. (2016). Link Prediction in Dynamic Networks Using Graphlet. In P. Frasconi, N. Landwehr, G. Manco, & J. Vreeken (Eds.), Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I (pp. 394–409).

Questions?

sandra.mitrovic@kuleuven.be

KU LEUVEN