# SubRank: Subgraph Embeddings via a Subgraph Proximity Measure

Oana Balalau[1]* and Sagar Goyal[2]*

[1] Inria and École Polytechnique, France
oana.balalau@inria.fr
[2] Microsoft, Canada
sagoya@microsoft.com

**Abstract.** Representation learning for graph data has gained a lot of attention in recent years. However, state-of-the-art research is focused mostly on node embeddings, with little effort dedicated to the closely related task of computing subgraph embeddings. Subgraph embeddings have many applications, such as community detection, cascade prediction, and question answering. In this work, we propose a subgraph to subgraph proximity measure as a building block for a subgraph embedding framework. Experiments on real-world datasets show that our approach, SubRank, outperforms state-of-the-art methods on several important data mining tasks.

**Keywords:** subgraph embeddings · personalized pagerank

## 1 Introduction

In recent years we have witnessed the success of graph representation learning in many tasks such as community detection [19, 8], link prediction [10, 20], graph classification [3], and cascade growth prediction [13]. A large body of work has focused on node embeddings, techniques that represent nodes as dense vectors that preserve the properties of nodes in the original graph [9, 5]. Representation learning of larger structures has generally been associated with embedding collections of graphs [3]. *Paths, subgraphs and communities embeddings* have received far less attention despite their importance in graphs. In homogeneous graphs, subgraph embeddings have been used in community prediction [8, 1], and cascade growth prediction [13, 6]. In heterogeneous graphs, subgraphs embedding have tackled tasks such as semantic user search [14] and question answering [4].

Nevertheless, the techniques proposed in the literature for computing subgraph embeddings have at least one of the following two drawbacks: *i*) they are supervised techniques and such they are dependent on annotated data and do not generalize to other tasks; *ii*) they can tackle only a specific type of subgraph.

* Part of this work was done while the authors were at Max Planck Institute for Informatics, Germany.

**Approach.** In this work, we tackle the problem of computing subgraph embeddings in an unsupervised setting, where embeddings are trained for one task and will be tested on different tasks. We propose a *subgraph embedding method based on a novel subgraph proximity measure*. Our measure is inspired by the random walk proximity measure Personalized PageRank [11]. We show that our subgraph embeddings are *comprehensive* and achieve competitive performance on three important data mining tasks: community detection, link prediction, and cascade growth prediction.

**Contributions.** Our salient contributions in this work are:
- We define a novel subgraph to subgraph proximity measure;
- We introduce a framework that learns comprehensive subgraphs embeddings;
- In a thorough experimental evaluation, we highlight the potential of our method on a variety of data mining tasks.

## 2 Related Work

**Node embeddings.** Methods for computing node embeddings aim to represent nodes as low-dimensional vectors that summarize properties of nodes, such as their neighborhood. The numerous embedding techniques differ in the computational model and in what properties of nodes are conserved. For example, in matrix factorization approaches, the goal is to perform dimension reduction on a matrix that encodes the pairwise proximity of nodes, where proximity is defined as adjacency [2], $k$-step transitions [7], or Katz centrality [16]. Random walk approaches have been inspired by the important progress achieved in the NLP community in computing word embeddings [15]. These techniques optimize node embeddings such that nodes co-occurring in short random walks in the graph have similar embeddings [18, 10]. Another successful technique is to take as input a node and an embedding similarity distribution and minimizes the KL-divergence between the two distributions [19, 20].

**Subgraph embeddings.** A natural follow-up question is how to compute embeddings for larger structures in the graph, such as paths, arbitrary subgraphs, motifs or communities. In [1], the authors propose a method inspired by ParagraphVector [12], where each subgraph is represented as a collection of random walks. Subgraph and node embeddings are learned such that given a subgraph and a random walk, we can predict the next node in the walk using the subgraph embedding and the node embeddings. The approach is tested on link prediction and on community detection, using ego-networks to represent nodes. In [13], the authors present an end-to-end neural framework that given in input the cascade graph, predicts the future growth of the cascade for a given time period. A cascade graph is sampled for a set of random walks, which are given as input to a gated neural network to predict the future size of the cascade. [6] is similarly an end-to-end neural framework for cascade prediction, but based on the Hawkes process. The method transforms the cascade into diffusion paths, where each path describes the process of information propagation within the observation time-frame. Another very important type of subgraph is a community and in [8] community embeddings are represented as multivariate Gaussian distributions.

**Graph embeddings.** Given a collection of graphs, a graph embedding technique will learn representations for each graph. In [3], the authors propose an inductive framework for computing graph embeddings, based on training an attention network to predict a graph proximity measure, such as graph edit distance. Graph embeddings are closely related to graph kernels, functions that measure the similarity between pairs of graphs [21]. Graph kernels are used together with kernel methods such as SVM to perform graph classification [22].

## 3 Feature Learning Framework

**Preliminaries.** PageRank [17] is the stationary distribution of a random walk in which, at a given step, with a probability $\alpha$, a surfer teleports to a random node and with probability $1-\alpha$, moves along a randomly chosen outgoing edge of the current node. In Personalized PageRank (PPR) [11], instead of teleporting to a random node with probability $\alpha$, the surfer teleports to a randomly chosen node from a set of predefined seed nodes. Let $Pr(u)$ be the PageRank of node u and $PPR(u,v)$ be the PageRank score of node $v$ personalized for seed node $u$.

**Problem statement.** Given a directed graph $G=(V,E)$, a set of subgraphs $S_1, S_2, \cdots, S_k$ of $G$ and an integer $d$, compute the $d$-dimensional embeddings of the subgraphs.

### 3.1 Subgraph Proximity Measure

We define a subgraph proximity measure inspired by Personalized PageRank. Let $S_i$ and $S_j$ be two subgraphs in a directed graph $G$. Their proximity in the graph is:

$$px(S_i, S_j) = \sum_{v_i \in S_i} PR_{S_i}(v_i) \sum_{v_j \in S_j} PR_{S_j}(v_j) \cdot PPR(v_i, v_j), \tag{1}$$

where $PR_{S_i}(v_i)$ represents the PageRank of node $v_i$ in the subgraph $S_i$, and $PPR(v_i, v_j)$ the PageRank of node $v_j$ personalized for node $v_i$ in the graph $G$.

When considering how to define proximity between subgraphs, our intuition is as follows: important nodes in subgraph $S_i$ should be close to important nodes in subgraph $S_j$. This condition is fulfilled as PageRank will give high scores to important nodes in the subgraphs and Personalized PageRank will give high scores to nodes that are "close" or "similar". We note that our measure is a similarity measure, hence subgraphs that are similar will receive a high proximity score. We choose the term *proximity* to emphasis that our measure relates to nearness in the graph, as it is computed using random walks.

We can interpret Eq. 1 using random walks, as follows: Alice is a random surfer in the subgraph $S_i$, Bob is a random surfer in the subgraph $S_j$, and Carol is a random surfer in graph $G$. Alice decides to send a message to Bob via Carol. Carol starts from the current node Alice is visiting ($PR_{S_i}(v_i)$) and she will reach a node $v_j \in S_j$ with probability $PPR(v_i, v_j)$. Bob will be there to receive the message with probability $PR_{S_j}(v_j)$.

**Normalized proximity.** Given a collection of subgraphs $S = \{S_1, S_2, \cdots S_k\}$, we normalize the proximity $px(S_i, S_j), \forall j \in 1, k$ such that it can be interpreted as a probability distribution. The normalized proximity for a subgraph $S_i$ is:

$$\hat{px}(S_i, S_j) = \frac{px(S_i, S_j)}{\sum_{S_k \in S} px(S_i, S_k)} \tag{2}$$

**Rank of a subgraph.** Similarly to PageRank, our proximity can inform us of the importance of a subgraph. The normalized proximity given a collection of subgraphs $S_1, S_2, \cdots S_k$ can be expressed as a stochastic matrix, where each row $i$ encodes the normalized proximity given subgraph $S_i$. The importance of subgraph $S_i$ can be computed by summing up the elements of column $i$.

**Sampling according to the proximity measure.** Given a subgraph $S_i$ in input, we present a procedure for efficiently sampling $px(S_i, \cdot)$ introduced in Eq. 1. We suppose that all the Pagerank vectors of the subgraphs $\{S_1, S_2, \cdots S_k\}$ have been precomputed. We first select a node $n_i$ in $S_i$ according to distribution $PR_{S_i}$. Secondly, we start a random walk from $n_i$ in the graph $G$ and we select $n_j$, the last node in the walk before the teleportation. Lastly, node $n_j$ may belong to several subgraphs $S_1^j, S_2^j \cdots$. We return a subgraph $S_j$ according to the normalized distribution $PR_{S_1^j}(n_j), PR_{S_2^j}(n_j), \cdots$. The procedure doesn't require computing the Personalized Pagerank vectors, which saves us $O(n^2)$ space. We shall use this procedure for computing embeddings, thus avoiding computing and storing the full proximity measure $px$.

### 3.2 Subgraph Embeddings via SubRank

Given a graph $G = (V, E)$ and set of subgraphs of $G$, $S = \{S_1, S_2, \cdots, S_k\}$, we learn their representations as dense vectors, i.e. as embeddings. We extend the framework in [20] proposed for computing node embeddings to an approach for subgraph embeddings. In [20], the authors propose to learn node embeddings such that the embeddings preserve an input similarity distribution between nodes. The similarities of a node $v$ to any other node in the graph are represented by the similarity distribution $sim_G$, where $\sum_{w \in V} sim_G(v, w) = 1$. The corresponding embedding similarity distribution is $sim_E$. The optimization function of the learning algorithm minimizes the Kullback-Leibler (KL) divergence between the two proximity distributions:

$$\sum_{v \in V} KL(sim_G(v, \cdot), sim_E(v, \cdot))$$

The authors propose several options for instantiating $sim_G$, such as Personalized PageRank and adjacency similarity. The similarity between embeddings, $sim_E$, is the normalized dot product of the vectors.

In order to adapt this approach to our case, we define the subgraph-to-subgraph proximity $sim_G$ to be the normalized proximity presented in Eq. 2. The embedding similarity $sim_E$ is computed in the same manner and the optimization function now minimizes the divergence between distributions defined on our input subgraphs, i.e. $sim_G, sim_E : S \times S \mapsto [0, 1]$. In our experimental evaluation we use this method, which we refer to as **SubRank**. We note that $sim_G$ will not be fully computed, but approximated using the sampling procedure presented in Section 3.1.

### 3.3 Applications

**Proximity of ego-networks.** Two very important tasks in graph mining are community detection and link prediction. Suppose Alice is a computer scientist and she joins Twitter. She starts following the updates of Andrew Ng, but also the updates of her friends, Diana and John. Bob is also a computer scientist on Twitter and he follows Andrew Ng, Jure Leskovec and his friend Julia. As shown in Figure 1, there is no path in the directed graph between Alice and Bob. A path-based similarity measure between nodes Alice and Bob, such as Personalized PageRank, will return similarity 0, while it will return high values between Alice and Andrew Ng and between Bob and Andrew Ng. An optimization algorithm for computing node embeddings will have to address this trade-off, with a potential loss in the quality of the representations. Thus, we might miss that both Alice and Bob are computer scientists. To address this issue we capture the information stored in the neighbors of the nodes by considering ego-networks. Therefore in our work, we represent a *node $v$ as its ego network* of size $k$ (the nodes reachable from $v$ in $k$ steps). In Section 4, we perform quantitative analysis to validate our intuition.

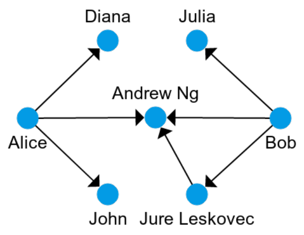

Fig. 1: Illustrative example for ego-network proximity.

**Proximity of cascade subgraphs.** In a graph, an information cascade can be modeled as a directed tree, where the root represents the original content creator, and the remaining nodes represent the content reshares. When considering the task of predicting the future size of the cascade, the nodes already in the cascade are important, as it very likely their neighbors will be affected by the information propagation. However, nodes that have reshared more recently the information are more visible to their neighbors. When running PageRank on a directed tree, we observe that nodes on the same level have the same score, and the score of nodes increases as we increase the depth. Hence, two cascade trees will have a high proximity score $\hat{px}$ if nodes that have joined later the cascades (i.e. are on lower levels in the trees) are "close" or "similar" according to Personalized Pagerank. In Section 5, we perform quantitative analysis and we show that our approach gives better results than a method that gives equal importance to all nodes in the cascade.

## 4 Feature Learning for Ego-networks

**Datasets.** We perform experiments on five real-world graphs, described below. We report their characteristics in Table 1.

- Citeseer[3] is a citation network created from the CiteSeer digital library. Nodes are publications and edges denote citations. The node labels represent fields in computer science.
- Cora[3] is also a citation network and the node labels represent subfields in machine learning.
- Polblogs[4] is a directed network of hyperlinks between political blogs discussing US politics. The labels correspond to republican and democrat blogs.
- Cithep[5] is a directed network of citations in high energy physics phenomenology. The network does not have ground-truth communities.
- DBLP[5] is a co-authorship network where two authors are connected if they published at least one paper together. The communities are conferences in which the authors have published.

| **Dataset** | Type | $\|V\|$ | $\|E\|$ | $\|L\|$ |
|---|---|---|---|---|
| Citeseer | Citation | 3.3K | 4.7K | 6 |
| Cora | Citation | 2.7K | 5.4K | 7 |
| Polblogs | Hyperlink | 1.4K | 19K | 2 |
| Cithep | Citation | 34K | 421k | 1 |
| DBLP | Co-authorship | 66K | 542K | 20 |

Table 1: Dataset description: type, vertices $V$, edges $E$, node labels $L$.

[3] https://linqs.soe.ucsc.edu/data
[4] http://networkrepository.com/polblogs.php
[5] https://snap.stanford.edu/data

**Competitors.** We evaluate our method, SubRank, against several state-of-the-art methods for node and subgraph embedding computation. For each method, we used the code provided by the authors. We compare with:

- DeepWalk [18] learns node embeddings by sampling random walks, and then applying the SkipGram model. The parameters are set to the recommended values, i.e. walk length $t = 80$, $\gamma = 80$, and window size $w = 10$.
- node2vec [10] is a hyperparameter-supervised approach that extends DeepWalk. We fine-tuned the hyperparameters $p$ and $q$ on each dataset and task. In addition, $r = 10, l = 80, k = 10$, and the optimization is run for an epoch.
- LINE [19] proposes two proximity measures for computing two $d$-dimensional vectors for each node. In our experiments, we use the second-order proximity, as it can be used for both directed and undirected graphs. We run experiments with $T = 1000$ samples and $s = 5$ negative samples, as described in the paper.
- VERSE [20] learns node embeddings that preserve the proximity of nodes in the graph. We use Personalized PageRank as a proximity measure, the default option proposed in the paper. We run the learning algorithm for $10^5$ iterations.
- VerseAvg is a adaption of VERSE, in which the embedding of a node is the average of the VERSE embeddings of the nodes in its ego network.
- sub2vec [1] computes subgraph embeddings and for the experimental evaluation, we compute the embeddings of the ego networks. Using the guidelines of the authors, for Cora, Citeseer and Polblogs we select ego networks of size 2 and for the denser networks Cithep and DBLP, ego networks of size 1.

For the first four methods, node embeddings are used to represent nodes. For sub2vec, SubRank and VerseAvg, the ego network embedding is the node representation. The embeddings are used as node features for community detection and link prediction. We compute 128 dimensional embeddings.

**Parameter setting for SubRank.** We represent each node by its ego network of size 1. We run the learning algorithm for $10^5$ iterations. **Our code is public.**[6]

**Running time SubRank.** In the interest of space, we report only the time required by SubRank for computing ego network embeddings. We run the experiments on a Intel Xeon CPU E5-2667 v4 @ 3.20GHz, using 40 threads. The running times are as follows: 1m40s Citeseer, 1m26s Cora, 49s Polblogs, 19m39s for Cithep and 39m45s for DBLP.

## 4.1 Node Clustering

We assess the quality of the embeddings in terms of their ability to capture communities in a graph. For this, we use the k-means algorithm to cluster the nodes embedded in the $d$-dimensional space. In Table 2 we report the Normalized Mutual Information (NMI) with respect to the original label distribution. On Polblogs, SubRank has a low NMI, while on Citeseer and Cora it outperforms the other methods. On DBLP it has a comparative performance with VERSE.

[6] https://github.com/nyxpho/subrank

| | **Dataset** | | | |
|---|---|---|---|---|
| **Method** | Citeseer | Cora | Polblogs | DBLP |
| DeepWalk | 0.015 | 0.018 | 0.013 | 0.314 |
| node2vec | 0.023 | 0.100 | 0.013 | 0.336 |
| LINE | 0.084 | 0.208 | **0.448** | 0.284 |
| VERSE | 0.103 | 0.257 | 0.024 | **0.363** |
| VerseAvg | 0.125 | 0.310 | 0.318 | 0.360 |
| sub2vec | 0.007 | 0.004 | 0.001 | 0.001 |
| SubRank | **0.179** | **0.347** | 0.021 | 0.357 |

Table 2: Normalized Mutual Information (NMI) for node clustering.

### 4.2 Node Classification

Node classification is the task of predicting the correct node labels in a graph. For each dataset, we try several configurations by varying the percentage of nodes used in training. We evaluate the methods using the micro and macro $F1$ score, and we report the micro $F1$, as both measures present similar trends. The results are presented in Table 3. On Citeseer and Cora SubRank significantly outperforms the other methods. On Polblogs, SubRank performs similarly to the other baselines, even though the embeddings achieved a low NMI score. On DBLP, SubRank is the second best method.

### 4.3 Link prediction

To create training data for link prediction, we randomly remove 10% of edges, ensuring that each node retains at least one neighbor. This set represents the ground truth in the test set, while we take the remaining graph as the training set. In addition, we randomly sample an equal number of node pairs that have no edge connecting them as negative samples in our test set. We then learn embeddings on the graph without the 10% edges. Next, for each edge $(u, v)$ in the training or the test set, we obtain the edge features by computing the Hadamard product of the embeddings for $u$ and $v$. The Hadamard product has shown a better performance than other operators for this task [10, 20]. We report the accuracy of the link prediction task in Table 4. Our method achieves the best performance on 4 out of 5 datasets.

## 5 Feature Learning for Information Cascades

Given in input: *i*) a social network $G = (V, E)$, captured at a time $t_0$, *ii*) a set of information cascades $C$ that appear in $G$ after the timestamp $t_0$, and that are captured after $t_1$ duration from their creation, *iii*) a time window $t_2$, our goal is to predict the growth of a cascade, i.e. the number of new nodes a cascade acquires, at $t_1 + t_2$ time from its creation. Note that given a cascade

| | **% of labelled nodes** | | | |
|---|---|---|---|---|
| **Method** | 1% | 5% | 10% | 20% |
| DeepWalk | 20.95 | 19.98 | 22.26 | 26.90 |
| node2vec | 31.20 | 31.11 | 28.24 | 30.15 |
| LINE | 30.81 | 33.58 | 44.81 | 53.62 |
| VERSE | 24.48 | 31.61 | 41.52 | 51.66 |
| VerseAvg | 29.46 | 38.51 | 46.12 | 55.77 |
| sub2vec | 18.23 | 20.17 | 22.04 | 22.79 |
| SubRank | **40.46** | **48.52** | **55.75** | **61.66** |

(a) $F1$ micro score for classification in Citeseer

| | **% of labelled nodes** | | | |
|---|---|---|---|---|
| **Method** | 1% | 5% | 10% | 20% |
| DeepWalk | 46.40 | 45.82 | 47.53 | 50.25 |
| node2vec | 46.40 | 44.26 | 46.55 | 50.16 |
| LINE | 44.46 | 46.28 | 57.54 | 61.74 |
| VERSE | 46.77 | 50.48 | 58.57 | 65.25 |
| VerseAvg | 53.11 | 57.83 | 65.75 | 70.14 |
| sub2vec | 25.55 | 46.38 | 46.42 | 46.42 |
| SubRank | **65.08** | **71.78** | **73.66** | **76.83** |

(b) $F1$ micro score for classification in Cora

| | **% of labelled nodes** | | | |
|---|---|---|---|---|
| **Method** | 1% | 5% | 10% | 20% |
| DeepWalk | 83.19 | 87.78 | 89.63 | 89.68 |
| node2vec | 85.83 | **88.91** | **89.93** | **90.52** |
| LINE | 73.50 | 87.50 | 89.70 | 89.34 |
| VERSE | 81.30 | 86.86 | 87.77 | 87.75 |
| VerseAvg | **87.12** | 88.77 | 86.26 | 88.42 |
| sub2vec | 51.55 | 51.20 | 51.23 | 51.42 |
| SubRank | 85.70 | 88.94 | 88.59 | 88.33 |

(c) $F1$ micro score for classification in Polblogs

| | **% of labelled nodes** | | | |
|---|---|---|---|---|
| **Method** | 1% | 5% | 10% | 20% |
| DeepWalk | 41.43 | 46.79 | 50.77 | 54.09 |
| node2vec | 42.61 | 48.00 | 51.51 | 54.92 |
| LINE | 25.02 | 33.93 | 37.96 | 41.31 |
| VERSE | 44.22 | 48.21 | 51.14 | 54.81 |
| VerseAvg | 45.41 | **51.69** | **55.22** | **58.46** |
| sub2vec | 6.00 | 7.71 | 8.85 | 9.60 |
| SubRank | **45.67** | 50.84 | 54.70 | 57.06 |

(d) $F1$ micro score for classification in DBLP

Table 3: $F1$ micro score for the classification task.

$c = (V_c, E_c) \in C$, we know that the nodes $V_c$ are present in $V$, however $c$ can contain new edges not present in $E$.

**Datasets.** We select for evaluation two datasets from the literature:

- AMiner [13] represents cascades of scientific citations. We use the simplified version made available by the authors[7]. The dataset contains a global citation graph and the cascades graphs. A node in a graph represents an author and an edge from $a_1$ to $a_2$ represents the citation of $a_2$ in an article of $a_1$. A cascade shows all the citations of a given paper. There are 9860 nodes in the global graph and 560 cascade graphs that are split into training, test and validation sets. The global network is based on citations between 1992 and 2002, while the training set consists of papers published from 2003 to 2007. Papers published in 2008 and 2009 are used for validation and testing. The cascade graphs are captured at the end of 1 year and we predict the increase in citations after 1 and 2 years.

[7] https://github.com/chengli-um/DeepCas

| | Dataset | | | | |
|---|---|---|---|---|---|
| **Method** | Cora | Citeseer | Polblogs | Cithep | DBLP |
| DeepWalk | 67.42 | 50.75 | 81.93 | 72.24 | 97.83 |
| node2vec | 67.04 | 59.54 | 82.76 | 75.58 | 98.56 |
| LINE | 71.49 | 61.48 | 83.72 | 84.85 | 98.26 |
| VERSE | 70.17 | 67.16 | **85.83** | 92.18 | **99.27** |
| sub2vec | 52.46 | 50.32 | 53.30 | 51.08 | 50.28 |
| SubRank | **80.10** | **82.10** | 82.70 | **94.10** | **99.30** |

Table 4: Accuracy for link prediction.

- Sina Weibo [6] consists of retweet cascades occurring on June 1, 2016 on the social network. Each node in the graph represent a Sina Weibo user, and an edge between $u_1$ and $u_2$ represent a retweet of $u_2$ by $u_1$. Each cascade corresponds to the retweets of one message. The global network is constructed by the union of the cascades occurring in the first half of the day, while the training, test and validation cascades are taken from the second half of the day. The cascades are captured after 1h from the initial post timestamp and we predict the increase in retweets in 1h, 2h and by the end of the day.

| | **AMiner** | | **Sina Weibo** | | |
|---|---|---|---|---|---|
| **Time period** | 1 year | 2 years | 1h | 2h | 1 day |
| DeepCas | 2.764 | 2.946 | 6.978 | 9.544 | 13.284 |
| DeepHawkes | 2.088 | **1.790** | **2.403** | **2.368** | **3.714** |
| VERSE | 2.313 | 2.181 | 3.580 | 4.243 | 5.862 |
| SubRank | **1.984** | **1.809** | 3.354 | 3.797 | 4.818 |

Table 5: Mean squared error (MSE) for predicted increase in cascade size.

**Competitors.** We compare SubRank with the following state-of-the-art methods for the task of predicting the future size of cascades:

- DeepCas [13] is an end-to-end neural network framework that given in input the cascade graph, predicts the future growth of the cascade for a given period. The parameters are set to the values specified in the paper: $k = 200$, $T = 10$, mini-batch size is 5 and $\alpha = 0.01$.
- DeepHawkes [6] is similarly an end-to-end deep learning framework for cascade prediction based on the Hawkes process. We set the parameters to the default given by the authors: the learning rate for user embeddings is $5 \times 10^{-4}$ and the learning rate for other variables is $5 \times 10^{-3}$.

- In addition, we consider the node embedding method VERSE [20], as one of the top-performing baseline in the previous section. The node embeddings are learned on the original graph and a cascade is represented as the average of the embeddings of the nodes it contains. We then train a multi-layer perceptron (MLP) regressor to predict the growth of the cascade.

**Parameter setting for SubRank.** We recall that our subgraph proximity measure requires the computation of PPR of nodes in the graph and the PR of nodes in the subgraphs. For this task, we consider the PPR of nodes in the global graph and the PR of nodes in the cascades. We obtain the cascade embeddings which are then used to train an MLP regressor. For both VERSE and SubRank we perform a grid search for the optimal parameters of the regressor.

We report the mean squared error (MSE) on the logarithm of the cascade growth value, as done in previous work on cascade prediction [13, 6] in Table 5. We observe that SubRank out-performs VERSE thus corroborating our intuition that nodes appearing later in a cascade should be given more importance. The best MSE overall is obtained by the end-to-end framework DeepHawkes which is expected as the method is tailored for the task. We note, however, that SubRank achieves the best results on AMiner.

## 6 Conclusion

In this work, we introduce a new measure of proximity for subgraphs and a framework for computing subgraph embeddings. In a departure from previous work, we focus on general-purpose embeddings, and we shed light on why our method is suited for several data mining tasks. Our experimental evaluation shows that the subgraph embeddings achieve competitive performance on three downstream applications: community detection, link prediction, and cascade prediction.

## References

1. Adhikari, B., Zhang, Y., Ramakrishnan, N., Prakash, B.A.: Sub2vec: Feature learning for subgraphs. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 170–182. Springer (2018)
2. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on World Wide Web. pp. 37–48. ACM (2013)
3. Bai, Y., Ding, H., Qiao, Y., Marinovic, A., Gu, K., Chen, T., Sun, Y., Wang, W.: Unsupervised inductive graph-level representation learning via graph-graph proximity
4. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. arXiv preprint arXiv:1406.3676 (2014)
5. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering **30**(9), 1616–1637 (2018)

6. Cao, Q., Shen, H., Cen, K., Ouyang, W., Cheng, X.: Deephawkes: Bridging the gap between prediction and understanding of information cascades. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 1149–1158. ACM (2017)
7. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM international on conference on information and knowledge management. pp. 891–900. ACM (2015)
8. Cavallari, S., Zheng, V.W., Cai, H., Chang, K.C.C., Cambria, E.: Learning community embedding with community detection and node embedding on graphs. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. pp. 377–386. ACM (2017)
9. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78–94 (2018)
10. Grover, A., Leskovec, J.: 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 (2016)
11. Haveliwala, T.H.: Topic-sensitive Pagerank: A context-sensitive ranking algorithm for Web search. TKDE **15**(4) (2003)
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196 (2014)
13. Li, C., Ma, J., Guo, X., Mei, Q.: Deepcas: An end-to-end predictor of information cascades. In: Proceedings of the 26th international conference on World Wide Web. pp. 577–586 (2017)
14. Liu, Z., Zheng, V.W., Zhao, Z., Yang, H., Chang, K.C.C., Wu, M., Ying, J.: Subgraph-augmented path embedding for semantic user search on heterogeneous social network. In: Proceedings of the 2018 World Wide Web Conference. pp. 1613–1622 (2018)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
16. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1105–1114. ACM (2016)
17. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the Web. Tech. rep., Stanford InfoLab (1999)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: 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 (2014)
19. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. pp. 1067–1077 (2015)
20. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference. pp. 539–548 (2018)
21. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. Journal of Machine Learning Research **11**(Apr), 1201–1242 (2010)
22. Zhang, Z., Wang, M., Xiang, Y., Huang, Y., Nehorai, A.: Retgk: Graph kernels based on return probabilities of random walks. In: Advances in Neural Information Processing Systems. pp. 3964–3974 (2018)