# Faster Clustering Using Non-Backtracking Random Walks

Brian Rappaport, Anuththari Gamage, Shuchin Aeron

Tufts University
Department of Electrical and Computer Engineering

August 4, 2017

## Abstract

This paper presents VEC-NBT, a variation of the unsupervised graph clustering technique VEC (Ding et al., 2016), which empirically improves upon the performance of the original algorithm in both speed and accuracy. VEC uses a novel application of the state-of-the-art word embedding model word2vec (Mikolov et al.) to embed a graph in Euclidean space via random walks on the nodes of the graph. In VEC-NBT, we propose using a non-backtracking random walk in lieu of the backtracking random walk used in VEC, which requires much shorter walks on the graph to obtain results with comparable or greater accuracy by virtue of its faster mixing rate compared to normal backtracking random walks.

## 1 Introduction

HELLO IT ME (Not clear whether we're talking about DSD yet so I'm leaving this be for the time being)

Clustering or classification is traditionally one of the main machine learning tasks, with applications practically everywhere one would want to use machine learning. We consider clustering as a set of elements, using the existence or strength of pairwise interactions between elements as the edges of a graph.

In this project we will explore efficient and reliable ways of clustering nodes into groups/communities based on pairwise (noisy with missing links) interactions (modeled as edges on a graph). Recently in [2] the authors propose to impose a metric on the nodes, referred to as diffusion state distance, or DSD, derived using random walks on the graph to derive a robust affinity measure between nodes followed by spectral clustering [1]. On the other hand, in another recent paper [5] the authors generate "sentences" of nodes by random walks on the graph as in word2vec [3] followed by embedding the sentences of nodes using a popular method, namely SGNS, for finding Euclidean word embeddings for NLP tasks [4].

In the first part of this project we want to numerically explore the connection between these two approaches. While the approach in [2] builds upon the limiting DSD, that can be computed in closed form, between nodes it is computationally expensive for large graphs (?? - read the paper). On the other hand the approach in [5] is computationally more attractive it being based on SGNS using SGD like methods that are known to scale well for large problem sizes.

# 2 Node Embedding via Random Walks on a Graph

Identifying community structures in a graph requires defining the 'similarity' between nodes of a graph, both in a local setting (intra-cluster node similarity) and a global setting (inter-cluster node similarity). The VEC algorithm proposed by Ding et al. [5] quantifies this similarity by considering the nodes of a graph as "words" contained in "sentences" formed by random walks on the graph. Once this word-sentence representation is obtained, identifying graph communities is akin to finding semantic/syntactic similarity between words in a language, which is often defined by the frequency with which word pairs or word groups occur together in sentences.

Starting at each node in the graph, the VEC algorithm performs random walks of fixed length and identifies the set of nodes which the initial node is most likely to encounter during a walk. This set forms the "sentence" to which the initial "word" belongs. This word-sentence representation of the graph is then transforms into Euclidean vectors via the state-of-the-art neural word embedding model word2vec [3]. The node clusters are then identified using a standard k-Means algorithm, which defines the similarity of nodes using the Euclidean distance between their vector representations.

## 2.1 Proposed Algorithm

# 3 Theory

## 3.1 Backtracking Random Walks

[Should compare mixing rates between BT/NBT and show how VEC-NBT is faster because of faster convergence]

[Show why VEC-NBT is more accurate] [really why tho?]

no realli

a møøse once bit my sister

See the løvely läkes

## 3.2 Non-Backtracking Random Walks

[section 6 of [1], http://www.cs.yale.edu/homes/spielman/561/2012/lect10-12.pdf, Meila and Shi (2001), more...]

# 4 Experimental Setup

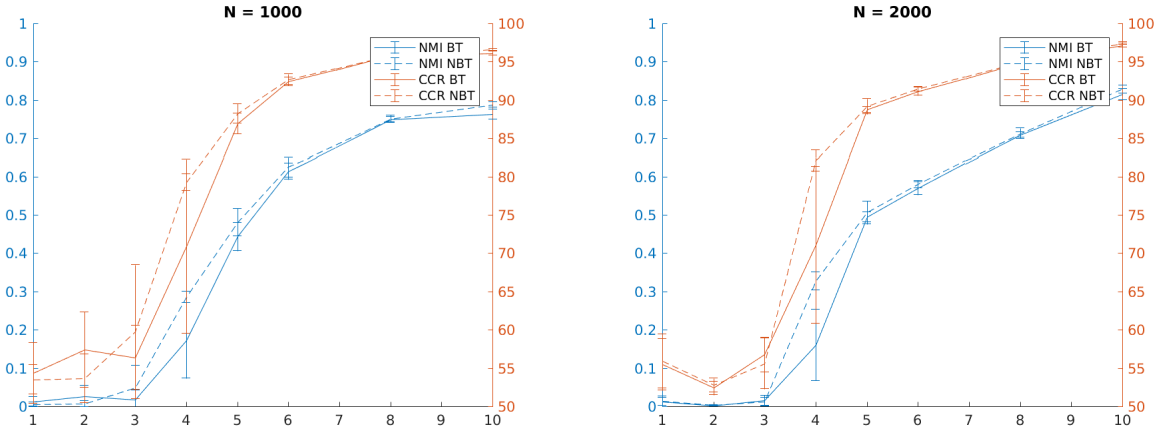
The Stochastic Block Model is a commonly used graph model for community detection. It builds off of the classical Erdős-Rényi $G(n, p)$ random graph model, wherein each edge of a graph with $n$ nodes is formed with probability $p$. In SBM (technically the planted partition model), $G(n, p)$ is additionally given $k$ clusters, and the probability an edge is formed within the cluster is $a$ while intercluster edges are formed with probability $b$. Clearly, $b$ should be lower than $a$ in order to form an assortative graph. In our work, we have elected to make $b$ proportional to $a$, so $b = \lambda a$ where $\lambda$ is between 0 and 1. Our model is represented as $S(n, k, p, \lambda)$.

[add hubs and spokes model description?- no, not enough data]

In order to generate sentences for use in the word2vec stochastic gradient descent algorithm, we need to perform random walks on the graphs. Since the graphs are unweighted, this is simply a matter of choosing at random among the neighbors of each node. We perform a certain number of walks starting from each node to a specific length. Here we also can set the walks to be reluctantly backtracking, where the walk will choose to return to a node immediately after leaving it only if it had no other possible choices. We will also not perform walks on any isolated nodes.

Once the sentences have been formed, we use an existing implementation of word2vec[3]. This takes a corpus of sentences (in this case, the random walks on the graph) and embeds them into $d$-dimensional space by means of stochastic gradient descent. Details on this algorithm can be found in [3].

# 5 Numerical Results




(a) 1000 data points

(b) 2000 data points

Figure 1: Accuracy of these algorithms, measured with two different metrics, correct classification rate and normalized mutual information. The x-axis c measures the sparsity of the graph, where an edge is made with probability c/N.

# 6 Discussion

# 7 Conclusion

# References

[1] Ulrike Von Luxburg. *A Tutorial on Spectral Clustering*. August. 2006.

[2] Mengfei Cao et al. "Going the Distance for Protein Function Prediction: A New Distance Metric for Protein Interaction Networks". In: *PLoS ONE* 8.10 (2013), pp. 1–12. ISSN: 19326203. DOI: 10.1371/journal.pone.0076339.

[3] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *CoRR* (2013), pp. 1–9. ISSN: 10495258. DOI: 10.1162/jmlr.2003.3.4-5.951. arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

[4] Omer Levy and Yoav Goldberg. “Neural Word Embedding as Implicit Matrix Factorization”. In: *Advances in Neural Information Processing Systems (NIPS)* (2014), pp. 2177–2185. ISSN: 10495258. DOI: 10.1162/153244303322533223. arXiv: 1405.4053. URL: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.

[5] Weicong Ding, Christy Lin, and Prakash Ishwar. “Node Embedding via Word Embedding for Network Community Discovery”. In: *CoRR* (2016), pp. 1–10. arXiv: 1611.03028. URL: http://arxiv.org/abs/1611.03028.