# CS 388, Fall 2018 Final Project: Link Detection in Political Networks

Diego Garcia-Olano & Eric Holgate

*Abstract*— **In this work we consider the task of link detection using a combination of natural language processing, network analysis and graph embedding methods over real-world politician networks. Networks for 219 Texas state and federal congressional members generated from a corpus of 50 thousand articles from various news sources in May 2015 are analyzed. We perform experiments comparing various baselines from each of the three methods and introduce a neural model which combines the representations from each baseline method. Factorization-, random walk- and deep learning-based graph embedding methods are compared for effectiveness and error analysis is used to look at patterns against party affiliation, and congressional level ( state vs federal ). The network data set will be made available and interactive networks and article snippets are currently available[1].**

## I. Introduction

Naturally occurring data sets in the form of academic authorship, biological and social networks [3] among many others are becoming more widely utilized. With concurrent advances in natural language processing and deep learning methods, more algorithms are currently being proposed to leverage these structures for various tasks.

Networks are constructed from the observed interactions between entities, be they authors, cells, politicians, etc and these interactions may be incomplete or inaccurate. The challenge often lies in identifying spurious interactions and predicting missing information. Link prediction refers to the task of predicting either missing interactions or links that may appear in the future in an evolving network[1]. Link prediction is used to predict probable friendships in social networks and can be used for recommendation in other settings.

As a motivating example for a potential use of link prediction on politician networks, consider the following scenario. If policy groups want to determine which politicians may be supportive of a given position and they have a politician C who already supports this position, they can compare all politicians they are considering in a pairwise fashion to see which have more of an affiliation or "link" with politician C through use of this methodology. In this way they could automatically compare which of two Democratic Texas Representatives from Austin, Eddie Rodriguez and Dawnna Dukes, is more likely to work with a conservative Republican Representative from East Texas, David Simpson.

Network analysis approaches to this task fall into the categories of similarity based, maximum likelihood based and probabilistic methods [8,9,10].

In addition to traditional network analysis techniques, methods which use the representation of graph nodes in vector space have gained traction from the broader research community [1,2]. These graph embedding techniques need to account for algorithm scalability, effects of dimensionality, features to be preserved, etc, and are divided into those based on factorization methods [7], random walks [4], and deep learning methods [6].

Additionally, within the field of natural language processing learning entity representation over documents via neural networks is an active field and the use of pre-trained language model to create embeddings for texts which can then be utilized in downstream tasks is widely used [11,12,13].

In this paper we study how methods using representations of entities derived from NLP, network analysis and graph embedding techniques along with combined approaches perform on the link detection task for a real world politician network consisting of 219 Texas congressional politicians [14]. Link detection in our setup concretely means given the articles and networks for politicians A and B, which of these two is more likely to have an affiliation or link with politician C given that politician's article set and network.

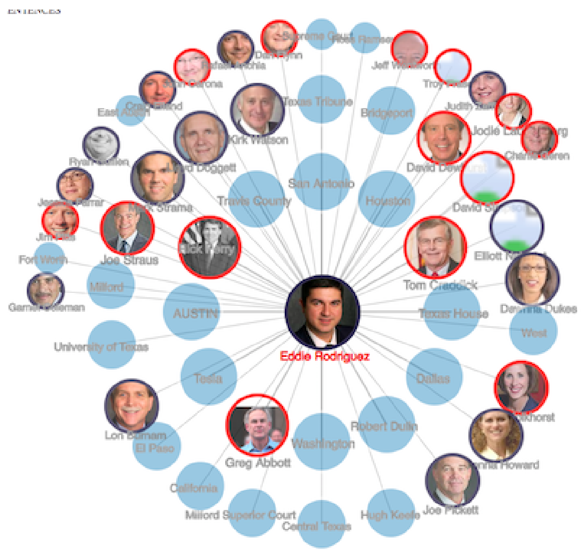


Fig. 1. Sample one hop network for Texas State House Democratic Representative Eddie Rodriguez, whose district is in East Austin

In Section II, we will describe the article corpus and subsequently derived politician network that we will use throughout the paper, and present specifics about the link detection problem in this context. In Section III, we will discuss background and related works. In Section IV we will discuss baseline models for the three entity representation techniques we consider along with how they perform. In Section V, we will discuss our combined models setup. In

[1]http://whoyouelect.com/texas/table-of-contents.html

Section VI will discuss overall results and do error analysis against additional information we have for the politicians we are modeling. Finally, in Section VII we will discuss future work and present concluding remarks.

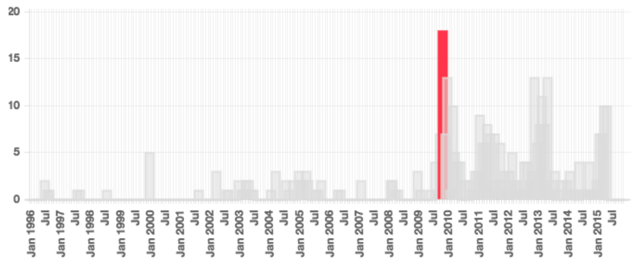


Fig. 2. time line of articles for Eddie Rodriguez

## II. Data set & Problem Formulation

### A. Who You Elect data set

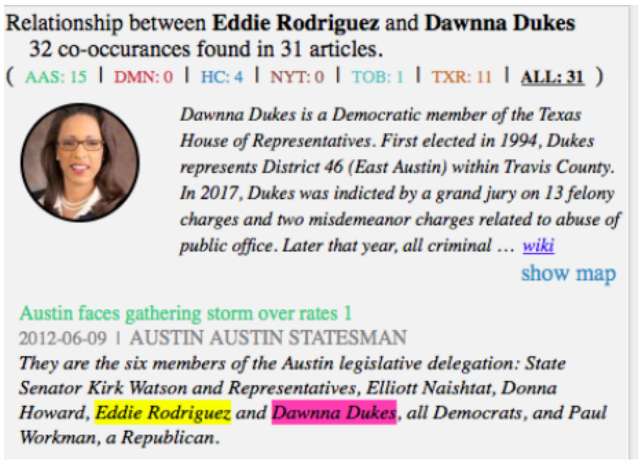
In a prior work [14], article sets for 219 State and Federal level congressional level Texan politicians actively in office in May 2015 were gathered independently for each politician from 6 online news sources: the Houston Chronicle, Dallas Morning News, Austin American Statesman, San Antonio Express, Texas Observer, Texas Tribune and the New York Times. Collectively, the corpus comprises approximately 50 thousand articles from between 2000 an 2015. The number of articles per politician ranges from 60 to 4,000 with most politicians having around 200 to 300 hundred. A modified open source Named Entity Recognition package [17] was then used to extract typed entities from each politician's article corpus. These types include `Politician, Person, Organization, Location, Bill and Misc`. For each politician, their articles were processed to gather entity co-occurrence statistics where each co-occurrence between entities was labeled as being within the "same sentence", "same paragraph" or "same article." This resulted in entity co-occurrence adjacency matrices for each politician, which can be visualized as graphs. Figure 1 shows one such politician network only considering immediate neighbors (i.e., "one hop")[2]. Figure 2 shows the temporal distribution of articles for that politician.

The prior work focused on automatic construction of undirected and weighted political networks and provided tools for exploring each network either (1) at the "one-hop" level which provides article snippets supporting links between nodes or (2) at the "extended" full graph level which provides links between all nodes in the graph and provides network analysis tools to do community detection, degree and power inspection. Figures 3 and 4 show examples of each network type. Table I shows the distribution of political party, level and congressional type for our data set. Appendix 1 shows maps depicting the breakdown by district over each level.

[2]View interactive version at http://www.whoyouelect.com/texas/explorer-view.html?show=20minor=1s=Eddie%20Rodriguez

| Party | Level | Type |
|---|---|---|
| Republican: 145 | State level: 181 | Representative: 186 |
| Democrat: 74 | Federal level: 38 | Senator: 33 |

TABLE I
Distribution of Politician Party, Level and Type in Dataset


Relationship between **Eddie Rodriguez** and **Dawnna Dukes**
32 co-occurances found in 31 articles.
( AAS: 15 | DMN: 0 | HC: 4 | NYT: 0 | TOB: 1 | TXR: 11 | **ALL: 31** )

*Dawnna Dukes is a Democratic member of the Texas House of Representatives. First elected in 1994, Dukes represents District 46 (East Austin) within Travis County. In 2017, Dukes was indicted by a grand jury on 13 felony charges and two misdemeanor charges related to abuse of public office. Later that year, all criminal ...* wiki
show map

Austin faces gathering storm over rates 1
2012-06-09 | AUSTIN AUSTIN STATESMAN
*They are the six members of the Austin legislative delegation: State Senator Kirk Watson and Representatives, Elliott Naishtat, Donna Howard, Eddie Rodriguez and Dawnna Dukes, all Democrats, and Paul Workman, a Republican.*

Fig. 3. Clicking on east Austin Texas House Representative Dawnna Dukes from within Eddie Rodriguez's one-hop network to see supporting articles

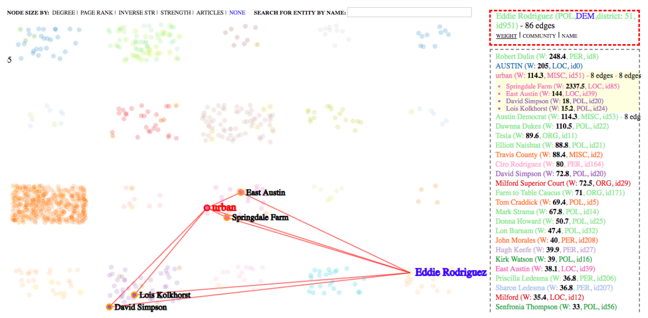
### B. Link detection task in Politician Networks

This work seeks to analyze the combined 219 networks, and considers the efficacy of using the smaller combined one-hop networks vs the larger combined extended networks.

Because the networks were generated independently of one another, the process of combining them into a single large network was complicated by the presence of some noise; for instance, separate networks contained nodes for Beto O'Rourke, Beto ORourke, Beto Rourke and hence required manual merging in those cases. In both the one-hop and extended cases, the combined networks contain `266,036 nodes`. There are `1,182,238 edges` in the *combined one-hop networks* whereas there are `101,820,341 edges` in the larger *combined extended networks*.

Filtering our combined network to only include the politicians of interest, we have a 219 by 219 count matrix which we use as the gold labels for our data set. The count matrix allows us to treat the task of link detection as a supervised task, since for three politicians A, B and C, we know the exact counts between A-C and B-C and hence know whether A or B has more of an affiliation with node C. Following this setup, we hold out 22 politicians and then create training, development and test splits of our politicians such that those 20 politicians only occur as the politicians we are making link detection predictions against in the test set ( i.e., politician C in our example above). We then take the set of all possible 3 politician combinations from the remaining 197 politicians and use 90% of it as training data and 10% as development data. We then create a test set where politician C comes from our held out set and sample unique combinations of A and B from the remaining 197 politicians.

In all of the three data sets, when dealing with articles and networks we remove any of politician C's articles or network edges that overlap with politician A or B's to keep our model from having apriori knowledge of the true decision[3]


NODE SIZE BY: DEGREE | PAGE RANK | INVERSE STR | STRENGTH | ARTICLES | NONE
SEARCH FOR ENTITY BY NAME:
5
East Austin
urban
Springdale Farm
Eddie Rodriguez
Lois Kolkhorst
David Simpson
WEIGHT | COMMUNITY | NAME

Eddie Rodriguez (POL,DEM,district: 51, id951) - 86 edges

Robert Dulin (W: **248.4**, PER, id8)
AUSTIN (W: **205**, LOC, id0)
urban (W: **114.3**, MISC, id51) - 8 edges - 8 edges
- Springdale Farm (W: **2337.5**, LOC, id85)
- East Austin (W: **144**, LOC, id39)
- David Simpson (W: **18**, POL, id20)
- Lois Kolkhorst (W: **15.2**, POL, id24)

Austin Democrat (W: **114.3**, MISC, id53) - 8 edg
Dawnna Dukes (W: **110.5**, POL, id22)
Tesla (W: **89.6**, ORG, id11)
Elliott Naishtat (W: **88.8**, POL, id21)
Travis County (W: **88.4**, MISC, id2)
Ciro Rodriguez (W: **80**, PER, id164)
David Simpson (W: **72.8**, POL, id20)
Milford Superior Court (W: **72.5**, ORG, id29)
Farm to Table Caucus (W: **71**, ORG, id171)
Tom Craddick (W: **69.4**, POL, id5)
Mark Strama (W: **67.8**, POL, id14)
Donna Howard (W: **50.7**, POL, id25)
Lon Burnam (W: **47.4**, POL, id32)
John Morales (W: **40**, PER, id208)
Hugh Keefe (W: **39.9**, PER, id27)
Kirk Watson (W: **39**, POL, id16)
East Austin (W: **38.1**, LOC, id39)
Priscilla Ledesma (W: **36.8**, PER, id206)
Sharon Ledesma (W: **36.8**, PER, id207)
Milford (W: **35.4**, LOC, id12)
Senfronia Thompson (W: **33**, POL, id56)

Fig. 4. Community detection over "extended" graph for Eddie Rodriguez

## III. Background and Related Work

We will first discuss general graph theory background, and the types of network analysis and graph embedding methods used for entity representation and then discuss related work using these methods towards the link detection task. The NLP baseline models we employ to create embeddings for each politician as a function of their articles are widely applied in the field so we leave detailed discussion of them to the reader [11].

### A. *Background and Network Analysis approaches*

A graph $G(V, E)$ is a collection of $V = \{v_1, ..., v_n\}$ vertices (a.k.a. nodes) and $E = \{e_ij\}_{i,j=1}^{n}$ edges. The adjacency matrix $S$ of graph $G$ contains non-negative weights associated with each edge: $s_{ij} \geq 0$. If $v_i$ and $v_j$ are not connected to each other, then $s_{ij} = 0$. The edge weight $s_ij$ is generally treated as a measure of similarity between the nodes $v_i$ and $v_j$; the higher the edge weight, the more similar the two nodes are expected to be.

Within the domain of network analysis techniques, similarity measurements can seen as focusing on global proximity or local proximity. *Local*, in this context, refers to both immediate neighbors of a given node, called first-order proximity, and the neighbors of the neighbors of a given called second-order proximity where the later compares the neighborhoods of two nodes for similarity.

Global proximities, such as Katz Index and rooted PageRank, attempt to preserve global asymmetric transitivity for directed graphs[7], but are computationally expensive and ill suited to larger networks in practice. The graph embedding methods we will consider attempt to handle global proximity and outperform these global proximity measures[7], so for graph-based baselines we focus on local proximity methods such as Common Neighbors [8] and Adamic-Adar.

Common Neighbors is a proximity measurement where for two nodes $v_i$ and $v_j$, the number of nodes linked directly to both $v_i$ and $v_j$ is used to measure their similarity. In other words the number of neighbors shared in common between nodes can be used to assess their similarity .

A weighted variant of Common Neighbors named Adamic-Adar assigns each neighbor a weight that is the reciprocal of the degree of the neighbor with the intuition being that neighbors of $v_i$ and $v_j$ with high degree are less important in evaluating the proximity between $v_i$ and $v_j$ than those of neighbors with lesser degree. Prior work [6,7] shows that Common Neighbors is superior to Adamic-Adar for the link detection task so we use it as our graph baseline method.

### B. *Graph embedding approaches*

Given a graph $G = (V, E)$, a graph embedding is a mapping $f : v_i \rightarrow y_i \in R^d \; \forall i \in [n]$ such that $d \ll |V|$ and the function $f$ preserves some proximity measure defined on graph $G$[1]. An embedding maps each node in $G$ to a low-dimensional feature vector and tries to preserve the connection strengths between nodes. For instance, an embedding preserving first-order proximity, given two node pairs $(v_i, v_j)$ and $(v_i, v_k)$ with edge weights $s_{ij} > s_{ik}$ will map $v_i$ and $v_j$ to points in the embedding space that will be closer to each other than the mappings of $v_i$ and $v_k$.

Embedding methods can fall into three broad categories:

- Factorization based ( such as *HOPE*) ,
- Random Walk based ( such as *node2vec* )
- Deep Learning based ( *SDNE* and *GCN* )

Factorization based algorithms represent the connections between nodes in the form of a matrix, usually a node adjacency matrix, but this could also be a node transition probability matrix, Katz similarity matrix, among others. The algorithms then factorize this matrix to obtain the embedding. Scalability is a major issue with this approach, whose time complexity is $O(|V|^2)$, and we focus on the High-Order Proximity preserved Embedding (HOPE) algorithm [7] which has been shown to be more effective at the link detection task than other factorization based methods [1,7]. HOPE preserves higher order proximity by minimizing $||S - Y_s Y_t{}^T||_F^2$ , where S is the similarity matrix. They represented each similarity measure as $S = M_g{}^1 M_l l$, where both $M_g$ and $M_l$ are sparse which enables the algorithm to use generalized Singular Value Decomposition to obtain the embedding efficiently.

Random walks have been used to approximate many properties of graphs and are especially useful when one can either only partially observe the graph, or the graph is too large to measure in its entirety [1]. The most used embedding technique using random walks on graphs to obtain node representations is *node2vec*[4]. The algorithm preserves higher-order proximity between nodes by maximizing the probability of observing the last $k$ nodes and the next $k$ nodes in a random walk centered at $v_i$. The model generates multiple "biased" random walks each of length $2k + 1$ and performs an optimization over sum of log-likelihoods for each random walk. A dot-product based decoder is used to reconstruct the edges from the node embeddings. The biased-random walks trade-off between breadth-first and depth-first

[3]Due to empirical difficulties with our graph embedding methods in general, we will need to revisit and assert this in experiments going forward

graph searches and produce more informative embeddings than prior approaches and enables *node2vec* to preserve community structure as well as structural equivalence between nodes. Although empirically effective, it lacks a clear objective function to articulate how to preserve the network structure and is prone to preserving only second-order and not higher proximity[6].

Deep learning methods research for learning representation embeddings from graphs has been an active field in recent years [18] and some recent promising approaches involve using deep auto-encoders for dimensionality reduction due to their ability to model non-linear structure in the data. Structural Deep Network Embedding (*SDNE*) [6] use deep autoencoders to learn embeddings which preserve the first and second order network proximities by jointly optimizing on these two proximities. The model consists of two parts: 1) an unsupervised part consisting of an auto-encoder aimed at finding an embedding for a node which can reconstruct its neighborhood, and 2) a supervised part which applies a penalty when similar vertices are mapped far from each other in the embedding space. *SDNE* has been shown to be better suited for the link detection task compared with other deep network methods so we focus our experiments on it. Another deep method of interest involves using Graph Convolutional Networks [5], whose results on the link detection task are comparable, but which has the added benefit of being able to integrate node features and will be explored in future work.

### C. Related Work

There has been quite a bit of literature involving either the use of network analysis or graph embedding techniques for the link detection task [1,6-10,20-24] Graph embeddings have been shown to effective at capturing the inherent dynamics of networks. Work predicting links from the learned node representations on publicly available collaboration and social networks[6,7] show links predicted using embeddings are more accurate than traditional similarity based link prediction methods on their datasets.

There seems, however, to have been relatively little work exploring the benefit of explicitly using article texts and derived networks to learn different embedding representations for the same entities and then combining those to train a model for the link detection task. Most work in the link detection area focuses on network topology structure and adding side information as labels on nodes and links, but does not leverage available text data in conjunction. For instance, Schlichtkrull et al. [25] propose an autoencoder link prediction model consisting of a "Relational-GCN" encoder producing latent feature representations of entities, and a tensor factorization decoder exploiting these representations to predict labeled edges over relational data. They introduce techniques to allow for parameter sharing and enforcement sparsity constraints, and use them to apply R-GCNs to multigraphs with large numbers of relations. The focus of the paper is leveraging network structure and features to create representations for link prediction, but do not leverage other representations for these entities as we are doing with our article set embeddings for politicians.

A perhaps more closely related work, Marcheggiani et al. [26] treats syntactic dependency trees over sentences as graphs and uses a stacked architecture to take individual word representations in a sentence that are each fed into a BiLSTM whose output is sent into a graph convolutional network (GCN) which then also leverages the syntactic dependency tree for the sentence to create a sentence embedding to train a classifier for semantic role labeling of predicates. Although similar machinery is being used in both works, i.e., the use of pre-trained language model representations and graph embeddings, the task and architecture vary significantly. Most importantly, the graphs they are dealing with are trees with few nodes and edges. In our case, the networks contain 250 thousand nodes and 100 million edges.

To the best of our knowledge, we could not find our particular problem formulation having been previously attempted.

## IV. Baseline Models

In constructing baselines for this task, we hypothesize two primary and useful sources of signal: (i) the set of articles in which a given politician appears and (ii) the social network of the politician (as defined in Section II). In this section, we present baselines for both sources of signal as well as baselines which utilize a combination of the two.

### A. Article Baseline

We begin by constructing a simple baseline using the article texts as the only source of signal. This approach is motivated by the fact that, due to the nature of political reporting as being politician- and/or issue- focused, we can approximate a politician's political identity through the way in which they are represented in newstext. This is exactly analogous to the way an individual may make conclusions from a number of political news articles to form a conceptualization of a candidate when contemplating their vote, or a politician when (re-)evaluating their level of support.

As a first pass, we construct rough *politician embeddings* through use of a pre-computed corpus of GloVe embeddings [28]. We treat the politician at the document-level (i.e., concatenate all the article texts in which that politician appears) and, withholding stopwords, compute the sum of the embeddings of the words in the document and normalize by word-count. However, to avoid unintentionally passing knowledge of the gold labels to our model, in each example we omit articles that contain both PolA and PolC or both PolB and PolC.

We then utilize these politician embeddings to compute cosine similarity scores between both answer candidates (PolA and PolB) and the target (PolC). Our model then predicts the candidate with the higher cosine similarity. With these cosine scores as the only predictors, we report an accuracy score of 62.22.

## *B. Network analysis Models*

We use the Common Neighbors algorithm as described in Section III to establish baselines for how it does on our data set. We experiment over different types of networks ( one-hop separate vs extended combined ) and with only politician entities, a subset of all entities or all entities to see how it performs on our test set. Table II shows the accuracy results. We note that combining the extended networks effectively adds noise and reduces performance slightly when considering only politician entities (i) or entities which occur in at least 10 separate politician networks (ii), but using the full combined extended network with all entities results in the best accuracy for the task. It is important to note here that combining networks in effect turns them from undirected symmetric individual networks into a directed asymmetric network. For example, the article set and derived network for politician A may show they have 30 links to politician B whereas the independently and separately gathered article set and derived network for politician B could then show them to have 35 links with politician A. We keep the combined networks symmetric by only considering the larger of the two edges between two politicians since a cursory analysis showed that most links between nodes were within 5 of each other and this noise was a by product of the manner in which the data was originally gathered.

| Type | **(i)only Politicians** | **(ii)20k entities** | **(iii)280k entities** |
|---|---|---|---|
| One-hop | 0.714 | 0.725 | 0.726 |
| Extended | 0.706 | 0.712 | 0.736 |

TABLE II

COMMON NEIGHBORS ACCURACY ON TEST SET

## *C. Graph Embedding Models*

We now consider the graph embedding methods discussed in the prior section; the factorization based method *HOPE*, the random walk based method *node2vec* and the deep learning auto-encoder model *SDNE*. Additionally we consider a ”Zero Hop” method which simply takes the 219 x 219 adjacency count matrix for our politicians and treats each row as the representation for a given politician.

We leverage the `openNE` library[27] which contains implementations of *HOPE*, *node2vec*, and *SDNE*.

Initial approaches to the problem involved attempting to run the different graph embedding techniques with a small hyper parameter grid search ( mostly over learning rate and embedding dimension size ) among the individual politician networks (both “one-hop” and “extended” ) and then using these for our task. We had a global set of entity node ids, but there seems to be a problem of identification, because evaluation using the independently learned embeddings from different graphs gave roughly 50% accuracy which isn't particularly inspiring for binary classification. Additional analysis of the constituent entity representations within individual networks to assess if cosine similarities between constituent representations gave promising interpretable results lead us to consider the combined networks formulation. Individual “one-hop” networks are insufficient for determining internode distances since all nodes are only connected to the main politician by definition, and for individual “extended networks”, there is some signal between nodes, but its also more limited than results from the combined case.

Accuracy performance of the graph embedding techniques over combined networks technique are shown in Table III. For the table we use 128 dimension vector embeddings with learning rates of .001 and default model specific parameters set in the original papers .

| Type | **(i)only Politicians** | **(ii)20k entities** |
|---|---|---|
| Zero Hop | 0.735 | x .498 |
| HOPE | 0.718 | 0.682 |
| node2vec | 0.663 | x |
| SDNE | 0.686 | 0.612 |

TABLE III

The downside to the combined networks paradigm is run time and memory consumption are significantly increased. In our use case *node2vec* took magnitudes of time longer than HOPE or SDNE and in fact consistently hung on the 20k entities subset, taking well over a day, while *HOPE* and *SDNE* finished within a couple hours. That time performance difficulty and the observation from [1] that node2vec generally performs worse on the link detection task, lead us to not assess it for columns 2 and 3. Similarly the Zero Hop idea of entity representation breaks down once we move into the later columns because although it does a great job at representing the politicians in a low dimensional space (i.e., 219), using a 20,000 dimensional “embedding” does not make sense and is too noisy to be feasible. Experiments for the full 280k entity adjacency matrix are ongoing and left for future work.

Given the our findings, we performed parameter tuning over the HOPE and SDNE by varying learning rates [ .05, .025, .009, .0075, .005, .0025, .0009, .00075 ] and embedding dimension size [ 100, 128, 200, 220]. The best model accuracy we obtained via SDNE was 69.88% with a learning rate of .0075 and dimension size of 128 whereas with HOPE the best model was that with the default learning rate of .001 and dimension size 128 as well.

A very *important finding* that we realized late in our work was that effectively our graph embedding setup made it so test information could bleed into our training information. The combined adjacency count matrix that the embeddings are based on contain node and edge information of the entire graph so when determining polA and polB's relationship with polC, all the information needed to deterministically make a decision is there so theoretically a deep net with enough parameters should be able to learn it. Initially when the idea was to use separate networks to learn embeddings the idea was simply to remove instances of C in the networks of A and B before training, but when that produce poor results and using combined networks improved results, the issue of bad experimental design was missed.

We did a preliminary test of fixing the Zero Hop setting on a 219x219 combined extended network and observed that its performance fell by roughly 10% to give 59.8% accuracy. The combined models in the next section leverage this Zero Hop setting and the best performing SDNE embedding on the combined extended 219x219 setting.

## V. Combined Models

We consider two variants of combined models to determine if considering both article-based and graph-based features in a neural architecture yields performance gains. To keep the architecture of the model and objective function simple for our baselines, we re-conceptualize the training task as edge weight prediction. By training a simple feed-forward neural network (FFNN) to predict the weight of the edge between two nodes, we can simply compare predicted weights downstream to yield a prediction as to which two nodes are more similar (N.B., we have defined node-node similarity as edge weight).

We present a simple FFNN which feeds input through a dense ReLU activation layer and applies dropout before prediction. We explore two manners of combining the article-based and graph-based features (**Concat** and **Diff**, described below) and two kinds of graph-based features (**Zero Hop** and **SDNE**, as have been described in Section III.

**Concat.** In this variant, we pass the FFNN the concatenated article ($A$) and graph ($G$) features for both the candidate ($C$) and target ($T$): $(C_A, C_G, T_A, T_G)$.
**Diff.** In this variant, we instead include the notion of comparison directly by passing the difference between the concatenated of the candidate and the concatenated features of the target as input: $(C_A, C_G) - (T_A, T_G)$

We utilize a development set of 100k training and 10k testing examples to tune the learning rate ($\eta$). All models were trained over 20 epochs. We present the Zero Hop tuning results below (Table IV).

| $\eta$ | **Concat** | **Diff** |
|---|---|---|
| 1e-2 | 54.41 | 64.8 |
| 1e-3 | 58.35 | 64.44 |
| 1e-4 | 44.74 | 63.07 |
| 5e-3 | 55.93 | 64.9 |

TABLE IV
Accuracy from Combined Model using Zero Hop embeddings on 100k/10k development data

Based on these results, we consider $\eta \in 0.001, 0.005$ when tuning $\eta$ for the SDNE representations (Table VI):

| $\eta$ | **Concat** | **Diff** |
|---|---|---|
| 1e-3 | 61.80 | 57.72 |
| 5e-3 | 56.07 | 60.97 |

TABLE V
Accuracy from Combined Model using SDNE embeddings on development data

After comparing the tuning results in Table IV and Table V, we opt to maintain $\eta \in 0.001, 0.005$ in testing, resulting in a total of eight variations of models and hyperparameters. The results are presented in Table VII:

| **Graph Embed** | $\eta$ | **Concat** | **Diff** |
|---|---|---|---|
| Zero Hop | 1e-3 | 57.99 | 63.23 |
| Zero Hop | 5e-3 | 53.03 | 61.09 |
| SDNE | 1e-3 | 61.87 | 55.98 |
| SDNE | 5e-3 | 57.25 | 52.46 |

TABLE VI
Accuracy from Combined Model using Zero Hop and SDNE embeddings on full data

Interestingly, we observe a difference in performance between the *Concat* and *Diff* input variations. We note that performance with Zero Hop graph embeddings as input is higher when using the difference between the candidate representation and target representation, while graph input from SDNE performs better when the representations are concatenated.

This could be due to the fact that, in computing the representation of a node, SDNE already takes the other nodes in the neighborhood into account, meaning that there is a baked-in dependence between the candidate and target nodes (in both directions). Due to the nature of the SDNE algorithm, this dependency is not necessarily limited to a single dimension of the computed representation, and may be distributed throughout the graph embedding. In the case of the Zero Hop graph embeddings, while there is still a notion of baked-in interdependence, it is limited to a single dimension. For this reason, subtracting the politician representations may amplify the difference between the tensors, making them generally more diverse (and thus informative). We note no consistent preference for learning rate.

## VI. Error Analysis and Visualization

### A. Error Analysis

We perform error analysis on our test set using the prediction results derived from our SDNE graph embeddings. On a high level, our test set had 58.4% cases where politician B was more linked to politician C than politician A was to C and our model underestimated that tendency by predicting politician B 49.3% of the time. Table VII,VIII, and IX shows the break down of all politicians in our test cases, the 197 considered for polA and polB and the 22 held out for polC. These numbers are useful to rule that the performance of our model was tied to big discrepancy in our test data vs the dataset as a whole.

| Political Party | PolA | **PolB** | **PolC** |
|---|---|---|---|
| Republican | 85013 | 82861 | 74227 |
| Democrat | 40476 | 42628 | 51262 |
| % Repub | 67.7% | 66% | 59.2 % |

TABLE VII
Test Set Break of party comparisons

| LEVEL | POSITION | PARTY Political Party |
|---|---|---|
| STATEWIDE:161 | REPRESENTATIVE:168 | DEMOCRAT : 64 |
| FEDERAL : 35 | SENATOR : 28 | REPUBLICAN:132 |

TABLE VIII

TEST SET BREAK DOWN OF POL A AND POL B

| LEVEL | POSITION | PARTY Political Party |
|---|---|---|
| STATEWIDE:19 | REPRESENTATIVE:17 | DEMOCRAT : 9 |
| FEDERAL : 3 | SENATOR : 5 | REPUBLICAN:13 |

TABLE IX

TEST SET BREAK DOWN OF POL C

We highlight the following specific case types for what should be easier and more differ cases,

- 1. cases where PolA and PolB are same party and PolC is different (ie, hard cases = 31301 = 24.
- 2. PolA and PolB are different parties and hence PolC matches with one ( easiest cases = 55792 = 44.5%)
- 3. PolA, PolB, PolC have the same party (also hard = 38396 = 30.6%)

Accuracy is pretty similar in all three cases with the lowest performance being when A and B are the same party and C differs (case 1) which gives 68.9% accuracy. The other two cases both give have near 70.3%). Looking at how these numbers breakdown by further, when polA and polB differ in party, whoever matches party with C is more closely related 71% of time; so party usually trumps. When all parties match however, and either A-C or B-C agrees in LEVEL, predicting based on that LEVEL is right 68.3% of time; so level plays a secondary role to party. At this point in time although we have District information for each politician (ie, which district ID they represent ), those IDs are not aligned over the Federal House, State House or State Senate so they can not be easily compared ( see Appendix I ). Future work will leverage this area of interest.

### B. Visualization

To gain a better understanding of what the graph embeddings are learning, we looked into visualizing our SDNE 219 x 219 embeddings using standard dimensionality reduction techniques, PCA [15] and t-SNE [16] and here show the results for the former. In Figure 5, we see two projections. They both show how politician embeddings are projected into a 2-D space. The left of the two shows the politician's names and observe Eddie Rodriguez to see politicians near him in the space which includes fellow Austin Representative Dawnna Dukes. The right one shows the same politician points except colored by political affiliation with orange being Republican. Its interesting to note that in the latter there is no strict division between the two parties as we had expected. Figure 6 on the one hand, shows the division which lies if you consider Congressional level (left) where blue = "State level congress" or Position type (right) where blue = "Representative" as opposed to "Senator" where we can see the data points are very clearly separated. This behavior especially at the Congressional level was noted in looking at the data.

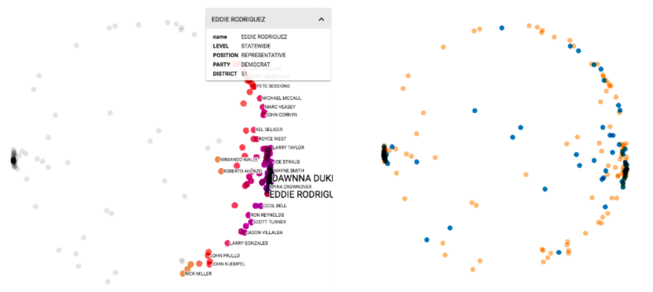


Fig. 5. (left) Eddie Rodriguez selected from (right) PCA projection of Politicians with party affiliation colored (orange = republican, blue = democrat)

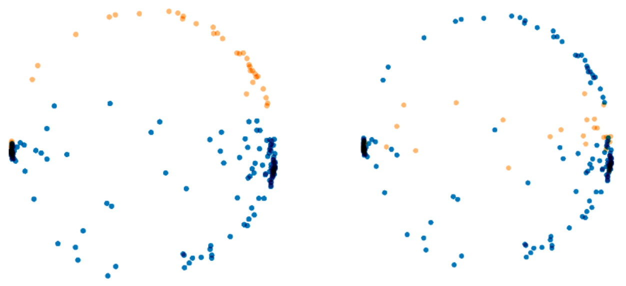


Fig. 6. (left)PCA projection of Politicians with Level of office ( state = blue, federal = orange ) and (right) Position type (orange = senator, blue = representative)

## VII. CONCLUSIONS & FUTURE WORK

There are many paths forward given our current results. First, due to our desire to focus on combining both article-based and graph-based representations, the article-based baseline presented here is overly simple. In immediate future work, we will seek to compute richer politician representations through use of sequenced collections of predicates describing the individual politicians (as opposed to using the context of the entire article). An initial approach to this end will involve training a separate neural network to predict the next predicate in a politician's sequence given the current sequence (a kind of "entity language model"). We will then experiment by using the weights in the hidden layer before the prediction layer as new politician representations.

Full article context may then be reintroduced to augment these representations, either in the computation of the politician representations themselves or downstream in the edge detection task.

We would also like to explore different formulations for our combined neural model. In its current state, it is framed as a regression task and this allows for consistency of the model, but its also possible that this problem might be better served by using framing it as a pairwise classification problem. In this setting the input to the model would be the joint representation [A; B; C], and the output would be +1 if A-C is closer, and -1 if B-C is closer. In either situation (either regression or classification), we may explore to having both datapoints ([A; B; C], +1) and ([B; A; C], -1) as samples in our training set, to "teach" our model that the relative positions of A and B are important in the input. The main difficulty with this approach is that you are not guaranteed consistency– if A is closer than B, and B is closer than C, then you want that A should be closer than C, but that need not always be true. Including constraints to enforce this consistency is possible by including all triplets, but is expensive.

Additionally, it would be interesting to explore multi-task learning for instance to see if predicting party affiliation or other information for entities aides in the task of link detection. To that end, we also have politician meta data we don't feed into our combined neural model and using GCNs to do so and create embeddings could be interesting.

Another aspect to consider is that our articles have temporal information that is not being exploited and this tie in nicely with related task of vote sequence prediction we hope to tackle once this model's accuracy is improved.

To conclude, in this paper we considered the task of link detection using a combination of natural language processing, network analysis and graph embedding methods over real-world politician networks. We show how well these methods perform apart on the networks and article sets in question and explore a method to combine them. Our findings provided us with insight into many additional future directions to consider. We presented a data set of 219 Texas political networks we hope others working in the area may find useful.

## REFERENCES

[1] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge Based Systems, vol. 151, pp. 7894, 2018.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, Representation learning on graphs: Methods and applications, IEEE Data Engineering Bulletin, vol. 40, no. 3, pp. 5274, 2017.

[3] M. Girvan and M. E. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences, vol. 99, no. 12, pp. 78217826, 2002

[4] 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. 855864.

[5] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907

[6] D. Wang, P. Cui, and W. Zhu, Structural deep network embedding, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 12251234.

[7] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, Asymmetric transitivity preserving graph embedding, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 11051114

[8] D. Liben-Nowell and J. Kleinberg.The link-prediction problem for social networks. Journal of the American society for information science and technology, 58(7):10191031, 2007.

[9] L.Lu, T.Zhou, "Link prediction in complex networks: A survey" ,Physica A: Statistical Mechanics and its Applications 390 (6) (2011) 11501170.

[10] M. AlHasan, M.J. Zaki, "A survey of link prediction in social networks," in: Social network data analytics, 2011, pp. 243275.

[11] J. Pennington, R. Socher, and C. D. Manning. "GloVe: Global Vectors for Word Representation," In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[12] M. Peters, L. Zettlemoyer, et al. "Deep contextualized word representations", in Proceddings of NAACL, 2018

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018

[14] D. Garcia-Olano, M. Arias and J. Larriba-Pey, "Automated Construction and Analysis of Political Networks via open government and media sources," in Proceedings of the First Workshop on Data Science for Social Good co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2016)

[15] I. T. Jolliffe, Principal component analysis and factor analysis, in: Principal component analysis, Springer, 1986, pp. 115128.

[16] L. v. d. Maaten and G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research, vol. 9, no. Nov, pp. 25792605, 2008

[17] MITIE: library and tools for information extraction https://github.com/mit-nlp/MITIE

[18] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (8) (2013) 17981828.

[19] Z. Yang, W. W. Cohen, and R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, in Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, pp. 4048, 2016.

[20] Hisano R "Semi-supervised graph embedding approach to dynamic link prediction." arXiv:1610.04351, arXiv preprint, 2016.

[21] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, and K. Tsuda, "Link Propagation: A Fast Semi-supervised Learning Algorithm for Link Prediction," in Proceedings of the 2009 SIAM International Conference on Data Mining. 2009, 1100-1111

[22] T. J. Lakshmi and S. D. Bhavani, "Link Prediction Measures in Various Types of Information Networks: A Review," 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, 2018, pp. 1160-1167.

[23] W. Yu, W. Cheng, C. Aggarwal, H. Chen, W Wang, "Link Prediction with Spatial and Temporal Consistency in Dynamic Networks", Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)

[24] s. Tsugawa, K. Kito, "Retweets as a Predictor of Relationships among Users on Social Media," in PLOS ONE 12(1): e0170279. (2017)

[25] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, "Modeling Relational Data with Graph Convolutional Networks," 2017

[26] D. Marcheggiani, I. Titov "Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling," Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 15061515

[27] "OpenNE: An open source toolkit for Network Embedding" https://github.com/thunlp/OpenNE

[28] "J. Pennington, R. Socher, C. D. Manning, "GloVe: Global Vectors for Word Representation," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1532-1543

## APPENDIX

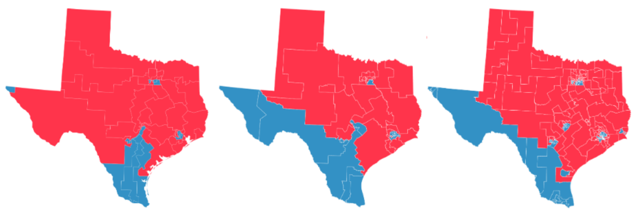


Fig. 7. Federal House, Texas Senate and Texas House districts by party affiliation in May 2015