18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Utilizing Covariates in Partially Observed Networks

David J. Marchette
Naval Surface Warfare Center,
18444 Frontage Rd, Suite 327,
Dahlgren, VA 22448

Elizabeth L. Hohman
Naval Surface Warfare Center,
18444 Frontage Rd, Suite 327,
Dahlgren, VA 22448

***Abstract*—In many applications of network analysis one observes the network of interest imperfectly. Inference on such networks requires one to either impute the unobserved information, or to be robust to the missing information. In many applications, such as social networks, brain connectomes, communications networks, etc. these graphs can be extremely large, which can make imputation problematic. Typically one has little to no control over the missing information. Additionally, there is often meta-data associated with the network, and incorporating these covariates can improve the inference and provide some level of robustness under the missing data. We will describe a family of spectral graph algorithms that allow one to utilize covariates in a natural way, and will illustrate the methodology on a large graph derived from Twitter, in which one observes most of the tweets containing geographic information (latitude and longitude of the device used to send the tweet) and builds the mentions graph, in which a directed edge indicates that a tweet from one user mentioned another user. Note that since we observe only those tweets with a location, we do not observe tweets from many of the users mentioned, and so do not have information about their outward edges or such covariates as their location, the languages they speak, etc. Note that in the case of the latter, we observe the meta-data partially as well – we observe the language of the tweets mentioning these users, but not those sent by them.**

We will describe a method to utilize the geographic information of the subset of users for which this is observed to perform inference on the overall graph, such as inferring missing edges and inferring the location of users whose devices do not provide this information.

I. Introduction

In many applications one observes information about relationships among entities that is best represented as a graph. The graph $G = (V, E)$ is a set of vertices (V) corresponding to the entities, and edges (E) – either directed or undirected, depending on the application – corresponding to the relationships. For this work we will consider only a single relationship, and simple graphs – no duplicated edges or edges between a vertex and itself.

In addition, we assume there are covariates – vectors of information associated with each vertex; thus the data we are interested in understanding and making inference about is $G_c = (V, E, X)$, where X is the matrix of d-dimensional covariates. The order of the graph is $|V|$ and the size is $|E|$. Typically, these graphs are very sparse, so $|E| << \binom{n}{2}$, but the covariates X are not.

A common class of algorithms for understanding graphs and making inferences about random graphs are those that utilize the spectrum of a graph. These all utilize some variant of the adjacency matrix of the graph, and compute eigenvectors or singular vectors (in the case of directed graphs) of this matrix. The most common matrices are the adjacency matrix itself, A which is the binary matrix whose (i, j) entry is 1 if and only if there is an edge from vertex i to vertex j (in the undirected case the matrix A is symmetric), and the Laplacian, which is $L = D - A$, where D is the diagonal matrix whose (i, i) entry is the degree of vertex i. One also sees a scaled version of the Laplacian $D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

One variant of these was used in [1] to predict unobserved edges. The basic idea is to use the singular vectors to represent the graph in $\mathbb{R}^d$ for some suitably chosen d, then predict edges between close vertices. If the embedding is in some way faithful to the edge probabilities, then close vertices will have a higher probability of edges between them.

Empirical evidence suggested that in the simplest case, where the vertices are drawn from a small set of k groups, and the edge probabilities are constant within and between groups (the blockmodel random graph), the embedded vertices would be distributed as a mixture of Gaussian distributions, with parameters depending on the group sizes and the $k \times k$ probabilities. This is proved in [2].

This suggests a simplistic approach to utilizing the covariates X. Embed the graph into $\mathbb{R}^d$ and adjoin X to these embedded points. Of course, this assumes that embedding and the covariates are commensurate. In [3] several methods are discussed in the introduction, and much of the literature on related methods is cited. The authors propose a variant of the spectral clustering method that utilizes the covariates directly. The authors refer to this algorithm as the CASC algorithm (for Covariate Assisted Spectral Clustering). It uses the leading eigenvectors of:

$$\widetilde{L}(h) = L_\tau + hXX^T. \tag{1}$$

Here, L_τ is a variant of the scaled Laplacian, where τ is a regularization parameter which has been observed to improve performance in sparse graphs. L_τ is defined as

$$D_\tau = \text{Diag}(\text{degrees(g)}) + \tau I \tag{2}$$

$$L_\tau = D_\tau^{-\frac{1}{2}} A D_\tau^{-\frac{1}{2}}. \tag{3}$$

We use the average degree as the regularization parameter τ as the authors recommend.

On its face it would appear that Equation (1) would not scale to large problems, since it appears to require the calculation of XX^T, an $n \times n$ matrix. However, the algorithm proposed never

actually computes this matrix. Instead, they use a Lanczos algorithm to compute the eigenvectors, which only requires calculations of the form $L_\tau v + hX(X^T v)$. See the paper for more information. Note that the form of Equation (1) is evocative of the random dot product graph formulation: the covariates are incorporated through their dot products. Although this connection appears tenuous, it motivated our thinking about the problem, and suggested the second approach.

Consider the case where the covariates X can be represented as a graph on the vertices. In the case we consider below, we are looking at using positions (geographic locations) along with the graph in order to infer a language distribution for users. It is reasonable to assume that the relevant information in the positions can be encoded in a nearest neighbor graph – the languages you speak will be correlated with the languages of people near you, and it isn't particularly informative to know just how near they are. Furthermore, it is likely that this correlation falls off with distance – knowing about people far away from you is not nearly as useful as those close to you. We propose the two variants of Equation (1) for use in this case. Let $nn_k(X)$ be the adjacency matrix for the k-nearest neighbors graph on the covariates X. The two methods proposed are essentially the same as what Equation (1) would be if we replaced the XX^T with a graph.

The first method uses the Laplacian and the nearest neighbor graph in place of the covariates:

$$L(g, X, \lambda) = \lambda L + (1 - \lambda) nn_k(X). \quad (4)$$

The second uses the random dot product embedding discussed in [1]:

$$DP(g, X, \lambda) = \lambda(A + D/(n-1)) + (1 - \lambda) nn_k(X). \quad (5)$$

These approaches are also related to the approach described in [4], [5] and [6]. These consider jointly embedding two (or more) sets of information into the same space. The difference is that each stream of information results in an embedded point, with two points associated with an object tending to embed close together. In the approach described here, the two streams of information result in a single point in the embedded space, with the two streams of information contributing to the position of the individual points and their spatial relationships to other observations.

We could weight the edges according to distance, for example using an exponential weighting scheme such as:

$$w_{ij} = e^{-\frac{d(x_i, x_j)}{\sigma}},$$

which would distinguish between high density regions like cities and lower density regions such as rural areas where the population tends to be more spread out. We feel that this would bias the results toward higher density regions, so we use the unweighted version in the experiment we discuss in Section II.

The weighted version could easily be incorporated in an application where the magnitude of the distances is important. In fact, it could be used to effectively eliminate the need for the k in the k-nearest neighbor graph. One would then construct a graph of those edges whose weight would be greater than a threshold τ, with this threshold chosen from practical considerations of the amount of information such a weighted edge could convey to the estimate. In some sense, this is what the CASC algorithm of Equation (1) does. Although it uses dot-product instead of distance, it is effectively a weighted complete graph, with the weights corresponding to the dot product similarity of the covariates. As mentioned above, this representation is definitely not the way one would wish to implement the algorithm, but it can be instructive to think about it in this light.

II. Application

We now turn to an application of the above algorithms to illustrate their performance on an inference task that uses a graph and covariates on the vertices of the graph. We consider the task of inferring the languages spoken by a Twitter user, given a graph indicating their contacts and the location from which they sent their tweets.

Twitter is a service that provides easy access to microblogs by millions of users. Users can send out short messages, which are seen by anyone who chooses to follow the user, as well as anyone who chooses to access the Twitter API[1]. Each microblog, or *tweet* consists of up to 140 characters, and can contain, in addition to the content, one or more links to other content on the web, and references to users. If a user is referred to (mentioned) in a tweet, that tweet is highlighted to the user. Thus the tweet can be thought of as being "sent" to the referenced user(s), even though it is also seen by all followers of the sender.

A. The Twitter Mentions Graph

The twitter mentions graph is a directed graph in which each vertex corresponds to a user and there is a directed edge between two users if the first mentions the second in a tweet. We will consider the set of tweets to be all tweets collected in April of 2014. Our collection strategy utilizes the Twitter streaming API to collect all tweets, up to a Twitter-imposed threshold, that contain a location within one of six rectangles. These rectangles, shown in Figure 1 are set up to roughly contain the six populated continents (although as can be seen in the Figure these are not strictly constrained by the continents' true boundaries).

For each of the continents we collected all the available tweets for the month of April.[2] The number of vertices and edges for each graph is shown in Table I. We also subset the mentions graph to the graph of mutual mentions – there is an edge between two vertices if each mentions the other at any time over the collection period.

As can be seen, there is a vast difference between the orders of the mentions digraph and the mutual mentions graph. This is because a vertex in the later must have a location within the

[1] https://dev.twitter.com/docs/api/streaming

[2] There were a couple of outages in the first few days of April due to power outages at our facility.

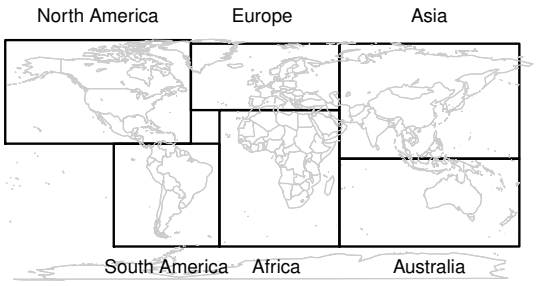


Fig. 1. Six regions covered by the Twitter collectors.

TABLE I
STATISTICS OF THE MENTIONS GRAPHS.

Continent	Mentions		Mutual Mentions	
	Order	Size	Order	Size
Africa	1,092,459	2,162,553	26,295	23,392
Australia	2,450,428	5,101,698	138,242	115,832
Asia	3,308,615	5,842,480	117,802	100,976
Europe	5,586,247	11,641,066	336,391	295,607
N. America	7,933,094	17,784,248	352,861	308,513
S. America	2,254,776	5,587,670	154,351	170,844

rectangle. A vertex can fail to be located within the rectangle either by being located outside the rectangle, or by failing to enable geo-location. If a user chooses not to geo-locate their tweets, they will not be collected by our sensors. Thus we have a clear selection bias in the edges we observe. This bias is unavoidable in the public API, which only allows access to a small percentage of available tweets. It is also common in many applications where one observes only a subset of edges, possibly because of a snowball sampling scheme or for availability reasons such as we have in our case.

Note also that there are non-human tweeters – robots (*bots* in the vernacular) that are set of to tweet out various information (weather information, newspaper headlines, quote-of-the-day, etc.) and most of these never mention any users, and many of them are never geo-located. However, users do mention these users (usually via retweeting their messages) and so these will show up in our mentions graph. The paper [7] describes a method for using the mentions graph to infer location, and so one might impute the location of these bots through algorithms such as these.

B. Geo-Location and Language

To illustrate the idea, we chose the following problem: given the mentions graph and positions for the observed vertices, infer the languages spoken by a vertex. The Twitter API provides a language code associated with each tweet. This is inferred from various information Twitter has about the tweet, as well as the content of the tweet itself. Assigning a language to a tweet is difficult and it is known that the algorithm used is imperfect, and it is possible (even within the 140 character limit) to send tweets containing more than one language, and these have been observed "in the wild". With that said, in this experiment we will take the language code as truth. Each user then has a table of the proportions of times they tweeted out a tweet that was assigned a given language code. We illustrate our methodology with two inference tasks:

1) Infer the proportions for a given user.
2) Infer the language that is used the most often by a user.

Each tweet we collect (with a small percentage of exceptions due to the vagaries of the Twitter geo-location process) has a geographic location coded as a latitude and longitude. This location is usually provided by the application (such as a smart phone) that sends the tweet. Thus it is usually the location given by a GPS unit. Occasionally it is provided by external sources – if the user clicks on the "tweet this link" icon on a web page, the page may assign a location to the tweet that is specific to the page. Also, users are free to spoof their latitude and longitude, and this is often done by researchers who are exploring Twitter as a source of data. Finally, people move around, sometimes quite extensively, which begs the question of what we mean by the (single) location of an individual computed from tweets taken throughout a given month.

The approach we have taken is to define a "home location" of the user to be the place from which they "tweeted the most". Specifically, we place a 50km disk around each tweet of the user, and the disk containing the most tweets is the "home location" of the user. This is essentially a two dimensional kernel estimator of the positions, and we take the maximum likelihood value as the "home location".

Since this is a covariate for our inference task, it is not too important how it is calculated or how accurate it is as a representation of where a person is. We use the values as a predictor for the value of interest, language, and the fact that it is noisy and less accurate for some users than others will be reflected in the variance of our estimator, as will be seen in the results below.

C. Results

We explore several methods for our inference tasks. All code is implemented using the R language [8].

g) You speak the same languages as those you mention. This uses only the mentions graph, and averages the tables from those who are mentioned (for which language data exists).
k) You speak the same languages as your friends. This is the same as above, but only considers the mutual mentions graph.
nn) You speak the language of your geographic neighbors. This ignores the mentions graph, and uses only the locations covariates. A k-nearest neighbors graph is constructed for $k = 10, 20, 50$ and in each case we proceed as above.
L) Laplacian embedding. Here we use Equation (4) to embed the users, using the mentions graph and the 50-nearest neighbors graph, and then uses the 10 nearest neighbors. The number 10 is arbitrary for illustrative

purposes. The embedding dimension is also 10, which is somewhat arbitrary but was chosen after a very preliminary analysis indicated that it was sufficient for the inference task. of the point in the embedded space to perform the inference.

DP) Dot product embedding. The same as the Laplacian version using Equation (5) instead.

The labels above correspond to the different methods, and are consistent with the labels in Figure 2.

It should be noted that the k-nearest neighbor graph can be computed using fast nearest neighbors algorithms (we use the version implemented in the `FNN` package in R [8], [9]). These algorithms can also be designed to return approximate nearest neighbors, which can speed up performance for larger problems, with a corresponding reduction in accuracy. For users that have no covariates (for example, users mentioned whose tweets are never collected for one of the reasons stated above), we add isolated vertices to the nearest neighbor graph. This provides a natural way to incorporate missing covariates. Since we have edge information in the graph, these vertices can still contribute to the inference.

Figure 2 depicts the language estimates for the 7 algorithms for the six continents. We depict the estimate errors as notched box plots, with the notches giving the rule-of-thumb significance for the box. In all cases we use Hellinger distance (computed using the R package `proxy` [10]) between the estimate and the true distribution:

$$d(x,y) = \sqrt{\sum_i \left(\sqrt{\frac{x_i}{\sum_j x_j}} - \sqrt{\frac{y_i}{\sum_j y_j}} \right)^2} \qquad (6)$$

The horizontal lines are an estimate of *random error*: we estimate the language by selecting a vertex at random from those with a language table. This is particularly useful as an indicator of continents for which the majority of users are monolingual, in which case one would expect an error of $\sqrt{2}$ (for example, Asia (Figure 2c) and Europe (Figure 2d)). We do not know if these two continents have this property because of the make-up of the tweeting population, something about the way the languages codes are assigned for these regions, the way the rectangular boundaries were set, or for some other reason specific to these populations, although in Figure 3 below we can see that these are the continents with the most diverse population of languages, which agrees to some extent with intuition. Africa, which is similarly diverse, has nearly as large a random error. We depict in red the error associated with the assignment of the major language to a user. This is computed simply as the proportion of times we make the wrong assignment.

Note that the mutual graph is always competitive, but it can only provide estimates for that small subset of users who have a cohort of geo-located friends amongst whom they tweet and mention each other. Ignoring these, the two embedding methods are always competitive, performing better than the other techniques (except possibly for Australia (Figure 2b) where the mentions graph g outperforms the dot product embedding, and all methods out perform the Laplacian). The embedding methods do not require mutual mentions, rather they utilize the partially observed, larger graph. Note also that Australia and North America have very similar characteristics. These are the two continents that are essentially bilingual (Indonesian and English, and English and Spanish, respectively), as indicated in Figure 3. Here we plot the proportion of users with a given primary language (a language used more than half the time by the user). Rare languages are removed to simplify the plots. We only consider users who have a primary language in this plot.

In all these examples we (again arbitrarily) set $\lambda = 0.5$, which weights the graph and the location covariates equally. It would be worth knowing the extent to which the choice of λ matters for this application. Looking at Figure 2, it seems clear that most of the time the graph information encoded in the mentions graph g and the location information encoded in the k-nearest neighbor graphs are comparable, and so one might suspect that our arbitrary choice is reasonable. We investigate this by optimizing the major language prediction over λ on a subset of $1,000$ users for each of the continents. This is depicted in Figure 4

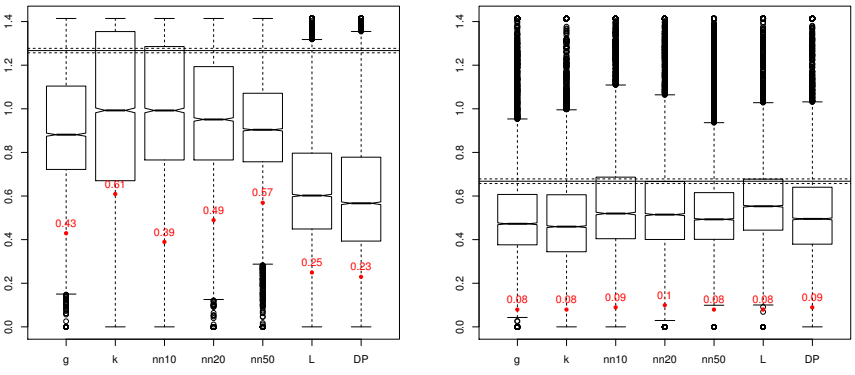
As we can see, the two datasets we called out as "monolingual", North America and Australia, have optimal values for λ around 0.4, but are pretty flat throughout the region. Further, these have a reasonable value at $\lambda = 0$ (which is equivalent to using only geography to determine language, which for these two regions is probably not unreasonable). The other data sets are much less geographical in nature, with the error value for $\lambda = 0$ being very far from optimal. South America is the only other region where geography is most important, according to this measure, while the other three place considerable weight on the mentions graph.

The above optimization was performed on a grid of width 0.1, using only 1000 observations (the same observations for each grid point/dataset pair), so obviously it could be improved upon. Also, using the guidance for setting h given in [3] might improve upon this approach. However, it does indicate that the optimal value of λ can be used to gain insight into the relative weighting of the graph and the covariates.

III. Conclusion

We have described two variants of the CASC algorithm of [3], and have illustrated their use on a problem of an incompletely observed graph with noisily observed covariates. The algorithms do a creditable job of predicting the major language of a user, and also beat or are competitive with algorithms that do not utilize the covariates (on the one hand) or the graph (on the other) in the examples we consider.

The CASC algorithm is shown in [3] to be consistent under a block model variant in which the graph and the covariates have distributions depending only on the group associate with the node. The algorithm variants we discussed in this paper are clearly both strongly related to the CASC algorithm and

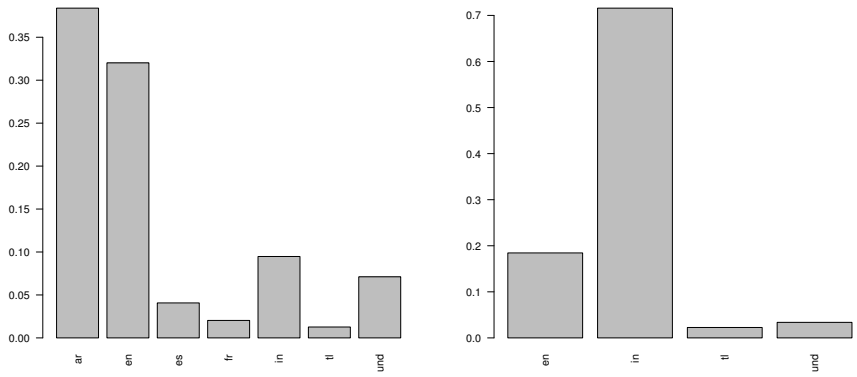


(a) Africa (b) Australia

(c) Asia (d) Europe

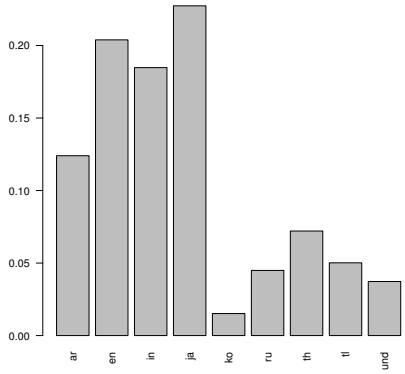
(e) NorthAmerica (f) SouthAmerica

Fig. 2. Language estimates for the six continents. The red dots/numbers are the errors for the assignment of primary language to the user. The horizontal solid/dashed lines are the median/significance for an estimate performed using a randomly chosen vertex. Each box corresponds to the estimates of $10,000$ vertices, chosen randomly from those that appear in the two graphs, for which location information is available.

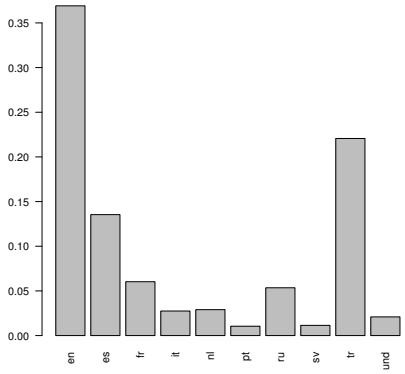


(a) Africa

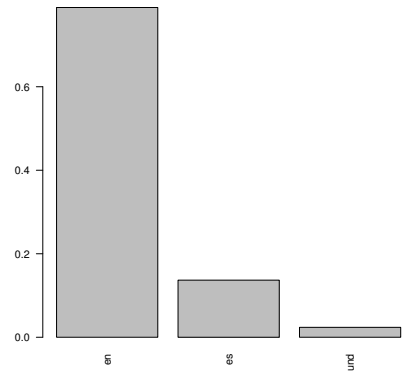
(b) Australia



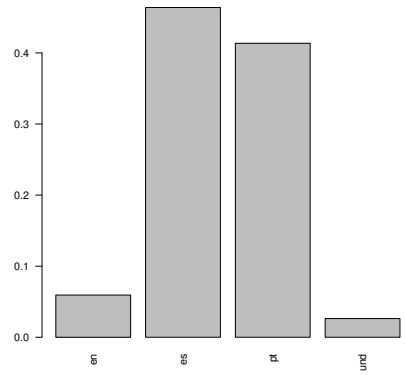
(c) Asia



(d) Europe



(e) NorthAmerica



(f) SouthAmerica

Fig. 3. Primary languages spoken by the users in each of the continents. These are the proportions of times each language is the primary language (used more than half time) of a user. Languages that are used by fewer than 1% of the users are removed from the plots. The language code "und" indicates that Twitter was unable to assign a language to the tweet.

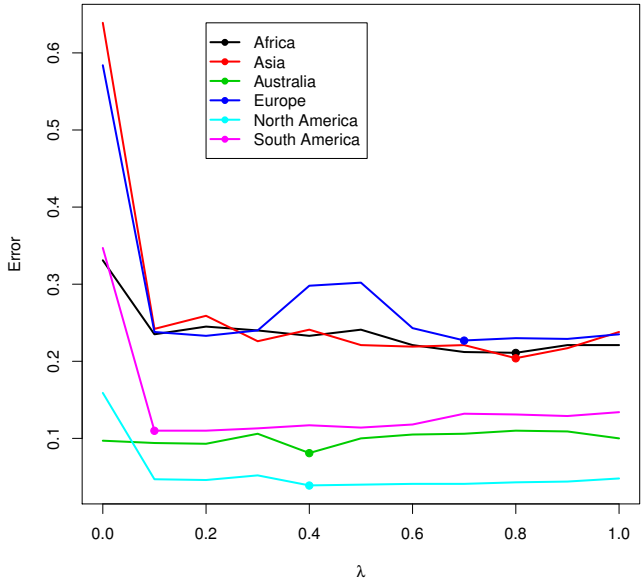


Fig. 4. Selection of the optimal values for λ (dots) for the six continents.

hence are likely to have the same property, although we have not yet verified this conjecture.

The CASC paper [3] also provides an algorithm for selecting h, which corresponds to our parameter λ. Our approach is more simplistic, and further research is warranted here. The objective function of optimizing the inference on the primary language is one choice, but if the desired inference is to obtain the full language-histogram this may be sub-optimal. One could easily insert the Hellinger distance into the algorithm in this case.

Looking at Figures 2 and 3, it seems that embedding methods perform best when more languages are spoken by the users. In some sense, in the simple cases where few languages are spoken almost anything can do a pretty good job of predicting language because the speakers tend to cluster into monolingual communities. In more rich and diverse cultures, there is indication that the embedding methods are needed in order to properly combine the two disparate streams of information.

Experiments on predicting the missing (unobserved) edges are ongoing. Preliminary indications are that neither the graph nor the locations are strong predictors of these missing edges, but it is challenging to design an experiment to test the performance of these algorithms given that we do not observe the edges. Experiments in which we artificially remove edges are under consideration.

These classes of spectral embeddings are extremely powerful, and the ability to jointly embed both graphs and vertex covariates provide a good class of algorithms for performing inference on these data. The algorithms utilize sparse spectral methods, and so can be applied to very large (sparse) graphs. We use fast nearest neighbor methods to construct the nearest neighbor graphs, and these can also scale well to large problems.

Acknowledgment

This work funded in part by the NSWC In-House Laboratory Independent Research (ILIR) program.

References

[1] D. Marchette and C. Priebe, "Predicting unobserved links in incompletely observed networks," *Computational Statistics and Data Analysis*, vol. 52, pp. 1373–1386, 2008.

[2] A. Athreya, V. Lyzinski, D. Marchette, C. Priebe, and D. Sussman, "A limit theorem for scaled eigenvectors of random dot product graphs," *Sankhya*, vol. to appear, 2015.

[3] N. Binkiewicz, J. T. Vogelstein, and K. Rohe, "Covariate assisted spectral clustering," *arXiv*, vol. arXiv:1411.2158 [stat.ML], 2014. [Online]. Available: http://http://arxiv.org/abs/1411.2158

[4] D. J. Marchette and J. L. Solka, "Fusion of disparate information through joint embedding," in *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, 2011.

[5] Z. Ma, D. J. Marchette, and C. E. Priebe, "Fusion and inference from multiple data sources in a commensurate space," *Statistical Analysis and Data Mining*, vol. 5, no. 3, pp. 187–193, 2012.

[6] C. E. Priebe, D. J. Marchette, Z. Ma, and S. Adali, "Manifold matching: Joint optimization of fidelity and commensurability," *Brazilian Journal of Probability and Statistics*, vol. 27, no. 3, pp. 377–400, 2013.

[7] D. J. Marchette, "A statistical analysis of a time series of twitter graphs," in *Proceedings of the Conference on Applied Statistics in Defense*, 2014.

[8] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: http://www.R-project.org/

[9] A. Beygelzimer, S. Kakadet, J. Langford, S. Arya, D. Mount, and S. Li, *FNN: Fast Nearest Neighbor Search Algorithms and Applications*, 2013, r package version 1.1. [Online]. Available: http://CRAN.R-project.org/package=FNN

[10] D. Meyer and C. Buchta, *proxy: Distance and Similarity Measures*, 2014, r package version 0.4-13. [Online]. Available: http://CRAN.R-project.org/package=proxy