LINK PREDICTION METHODS FOR GENERATING SPEAKER CONTENT GRAPHS

K. Greenfield, W. M. Campbell

MIT Lincoln Laboratory, Human Language Technology Group, Lexington, MA, USA
{kara.greenfield, wcampbell}@ll.mit.edu

ABSTRACT

In a speaker content graph, vertices represent speech signals and edges represent speaker similarity. Link prediction methods calculate which potential edges are most likely to connect vertices from the same speaker; those edges are included in the generated speaker content graph. Since a variety of speaker recognition tasks can be performed on a content graph, we provide a set of metrics for evaluating the graph's quality independently of any recognition task. We then describe novel global and incremental algorithms for constructing accurate speaker content graphs that outperform the existing k nearest neighbors link prediction method. We evaluate these algorithms on a NIST speaker recognition corpus.

Index Terms— speaker recognition, link prediction, network theory

1. INTRODUCTION

Speaker recognition based on vector methods has become the standard approach to text-independent speaker recognition. These methods take a speech recording and map it to a fixed size vector which is then used to perform recognition tasks. Common approaches use GMM supervectors [1] and iVectors [2]. Vector methods have an intuitive geometric interpretation as well as being amenable to standard machine learning, linear algebra, and Bayesian methods.

A recent development which leverages the vector-based approach is the use of speaker content graphs [3]. Speaker content graphs capture the manifold structure of a corpus of recordings by expressing similarity between close vectors. In a speaker content graph, vertices represent recordings and edges represent speaker similarity. The ideal speaker content graph would have edges only between vertices from the same speaker. In practice, edges representing false alarms are present and edges representing misses are absent.

Speaker content graphs have already proven to be an effective methodology in common speaker recognition and novel corpus level tasks. Prior work has explored the speaker comparison (1-1) problem using geodesics and multiple features sets, see [3] and [4]. Also, speaker retrieval using content graphs and a low-computation random walk approach was demonstrated in [5].

Although multiple applications for content graphs have been examined for content graphs, limited exploration of how to construct higher-quality graphs has been considered. Typically, two approaches appear in the semi-supervised learning literature for constructing similarity based graphs—epsilon neighborhoods of points and k-nearest neighbor (KNN) graphs [6]. For epsilon neighborhoods, connecting all vectors that are within a radius of a point, it was found in prior work [3] that the resulting graphs are very sensitive to the neighborhood chosen and produce unusual degree distributions. The alternate approach, using KNNs, has better properties, but still will always connect k neighbors regardless of whether these are the same speaker or not.

In this paper, we consider novel methods for graph construction. In standard approaches, edges are selected independently. We consider approaches that use prior knowledge about the expected global structure of the content graph. For instance, knowledge of the number of utterances per speaker should result in a target degree distribution for the graph. Another global constraint is that the graph structure should include cliques of connected vertices from the same speaker. The overall edge structure should be sparse. Using these global constraint ideas, we construct several algorithms which lead to improved quality graphs.

The outline of the paper is as follows. Section 2 describes background information on link prediction and speaker content graphs. In Section 3, we describe the baseline KNN method and present three new link prediction methods. In Sections 4 and 5, we explore the performance and sensitivity of the algorithms.

2. SPEAKER CONTENT GRAPHS

Content graphs are initially constructed based on methods from the semi-supervised learning literature [6, 7]. We start with a set of vectors $M = \{m_i\}$ obtained as a vector mapping of a corresponding set of speech signals. The vectors are contained in a manifold, and we create a graph G which reflects the local connectivity and distances of points on the manifold.

The procedure for constructing a speaker content graph is the following. First, each vertex v in the graph corresponds to a single vector m from a speech signal. We define weights for edges in the graph via a weight matrix $W = [W_{i,j}]$ where

$$W_{i,j} = \begin{cases} e^{-d^2(m_i,m_j)/\sigma^2} & \text{if an edge exists between } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

We consider the edge between i and j to be in the graph if $W_{i,j} \neq 0$.

The parameter σ controls the decay of the exponential function and corresponds to soft edge weight between 0 and 1. In our methods, we use an approximate KL divergence [1] for the distance in (1). In order for the graph to reflect the local neighborhood of v, we only connect it to a limited number of neighbors.

The truth graph is an instantiation of a speaker content graph with weight matrix $W = [W_{i,j}]$ where

$$W_{i,j} = \begin{cases} 1 & i \text{ and } j \text{ are from the same speaker} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Since originating from a given speaker is a transitive property, the truth graph is a union of mutually disjoint complete subgraphs.

3. LINK PREDICTION

Link prediction uses vertex similarity to compute which edges to include in a graph, either generating all of the edges or adding missing

*This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

edges to an existing graph. Traditionally link prediction has been performed for social networks or web pages [8]. Other efforts, such as [9] have focused more on clustering than quality of the link prediction. Our work proposes new methods that build off the current best method, the k nearest neighbors algorithm. We tailor our approach to produce graphs with many highly connected disjoint components.

3.1. Link Prediction Methods

We present four link prediction methods to generate content graphs. The first, KNN, was shown to be useful for speaker recognition by [3]. While KNN was a significant improvement over the ϵ graph baseline in [3] by better conforming to the true degree distribution, there is still room for improvement. A good content graph generation algorithm should produce graphs with the following criteria:

- High precision and recall of generated edges
- Realistic degree distribution
- High clustering coefficient

We measured the precision and recall by comparing edges contained in a generated graph to edges contained in the truth graph. We define a correct edge in the generated graph as an edge that is also in the truth graph. Specifically:

$$\text{Precision} = \frac{\text{Number of correct generated edges}}{\text{Number of generated edges}} \tag{3}$$

$$\text{Recall} = \frac{\text{Number of correct generated edges}}{\text{Number of edges in truth}}. \tag{4}$$

The degree of a vertex is the number of vertices that are adjacent to it and the degree distribution of a graph is the probability distribution of the degrees over all of the vertices in the graph. When comparing degree distributions, we looked at both the range of the distribution and the shape.

Clustering is a metric of edge transitivity. The clustering coefficient of a node is the proportion of pairs of its neighbors that are adjacent. The clustering coefficient of a vertex v is defined as follows:

$$c_v = \frac{2T(v)}{deg(v)(deg(v)-1)}, \tag{5}$$

where $T(v)$ is the number of triangles that contain v and $deg(V)$ is the degree of v. The clustering coefficient of a graph is the average clustering coefficient of its vertices [10]. A graph that is a union of complete subgraphs, such as the truth graph, will have a clustering coefficient of 1 while a randomly generated graph will have a clustering coefficient closer to 0.

By producing graphs that optimize the above parameters, we are implicitly generating sparse graphs with approximately one highly connected component per speaker. This allows us to generate close approximations to the truth graph.

3.2. k Nearest Neighbors

The KNN algorithm constructs content graphs in an incremental manner. As new speech data streams in over time, KNN dynamically updates the content graph to reflect the new data. We briefly describe two steps to make this an efficient process.

Suppose we have an existing data set $M = \{m_i\}$ with $|M| = n$, a new piece of data m_{n+1}, a content graph G, a list of indices, the closest k distances $\{D_{i,j}\}$ to each m_i, and a weight matrix W. Adding a new utterance to the graph is equivalent to appending a row and column to W, see Algorithm 1. The result is that the k closest points to m are inserted into D; for appropriate i, m is inserted into the distances and lists for m_i.

Algorithm 1 k Nearest Neighbors Link Prediction

Input: new vector to add, m_{n+1}
for $i = 1 \to k$ **do**
  Compute and store $d(m_{n+1}, m_i)$ and store a sorted list, $D_{n+1,i}$ and corresponding indices.
for $i = k+1 \to n$ **do**
  Compute $d(m, m_i)$ using an early out algorithm:
  Retrieve the furthest neighbor distance $D_{i,k}$ for m_i.
  for the dimensions in the input vector m_{n+1} **do**
    The current estimate of the distance is monotonically increasing.
    if the estimate is greater than $D_{i,k}$ and $D_{n+1,k}$ **then**
      Go to the next i
  Insert the point in the appropriate list.

3.3. Greedy Link Prediction

KNN implicitly assumes an approximately uniform degree distribution and explicitly assumes that each vertex has degree at least k. As neither of these assumptions are realized in actual data, we need an alternative algorithm that generates content graphs with realistic degree distributions.

Greedy link prediction uses a modified version of KNN's setup. Instead of storing the distances indexed by utterances, the distances are stored in a single vector D and sorted in increasing order, $D_1 \leq D_2 \leq ...D_n$. Unlike KNN, which added utterances to the graph incrementally, greedy link prediction requires knowledge of all utterances a priori. Greedy link prediction also requires a desired degree distribution that is realizable as a graph with the required number of vertices. Greedy link prediction uses the steps in Algorithm 2 to construct a content graph.

Algorithm 2 Greedy Link Prediction

for $i = 1 \to n$ **do**
  Add the edge associated with D_i to G unless doing so will make it impossible to attain the desired degree distribution. This is done when one of the following conditions is met.

- One or both of the vertices incident with the edge associated with D_i already has the maximum degree specified in the degree distribution.
- Incrementing the degree of one or both of the vertices incident with the edge associated with D_i will cause there to be too many vertices of high degree.
- Incrementing the degree of one or both of the vertices incident with the edge associated with D_i will cause there to be too few remaining vertices of low degree.

In experiments, we found that using the 100 closest distances from each utterance in D worked well. Doing this removes the guarantee that the truth graph can be recovered. However, the changes to the graph will be minimal, and we observed that the improved efficiency outweighs the slight degradation in graph quality.

3.4. Force-Clique Link Prediction

A perfect content graph is a union of cliques, where a clique is defined as a set of n pairwise adjacent vertices (i.e., the induced subgraph is complete). Force-clique link prediction improves existing content graphs by removing some edges and inserting others in order to make the graph closer to this ideal. Suppose we have an existing content graph G. A maximal clique in G is a clique that cannot be enlarged to a larger clique by adding additional vertices.

We note that since D only contains the k closest distances to m_i, we may not know the appropriate weight for every newly added edge. In this situation, the average weight is used in liu of the exact weight without noticable degradation in the overall quality of the graph. Our global force-clique algorithm is shown in Algorithm 3.

Algorithm 3 Global Force-Clique Link Prediction

for each maximal clique C **do**
  if $|C| \geq minsize$ **then**
    for each vertex v not in C **do**
      if at least x percent of the vertices in C are adjacent to v **then**
        Connect all vertices in C to v.
      else
        Disconnect all vertices in C from v.

Force-clique link prediction can also generate graphs in an incremental manner. Suppose we have an existing data set $M = \{m_i\}$ with $|M| = n$, a new piece of data m_{n+1}, a content graph G, a list of indices, the closest k distances $\{D_{i,j}\}$ to each m_i, and a weight matrix W. Adding a new utterance to the graph is equivalent to appending a row and column to W. Incremental force-clique link prediction uses the steps in Algorithm 4 to add m to G.

Algorithm 4 Incremental Force-Clique Link Prediction

Use any incremental link prediction algorithm to add m to G. Let v be the vertex corresponding to m.
for each maximal clique C **do**
  if $|C| \geq minsize$ **then**
    if at least x percent of the vertices in C are adjacent to v **then**
      Connect all vertices in C to v.
    else
      Disconnect all vertices in C from v.

Incremental force-clique link prediction will not always generate the same graph as global force-clique link prediction. Differences arise in cases where v would have been in a clique if it had been added to G prior to applying force-clique link prediction. After force-clique link prediction is applied, we don't know which edges were present in G, so we don't know which cliques v would have been a member of. The differences, if any, are minimal and localized; some of the differences result in the incrementally generated graph being more accurate. As such, the computation savings incurred in applying incremental force-clique link prediction are typically worthwhile.

In our experiments, the best value for $minsize$ in both the global and incremental variations of force-clique link prediction was consistently 5. If G has high precision, a slightly lower value can be used. This allows for the generated graph to have a higher recall, but will drastically reduce precision if done inappropriately. Unless the precision of G is extremely low and the average degree is expected to be high, larger values should not be used for $minsize$. Doing so constricts force-clique link prediction, so that it only modifies a few of the edges in G. A better option is to use a differently parameterized algorithm to generate G so that it has higher precision.

4. COMPARISON OF GENERATED CONTENT GRAPHS

Experiments were performed on the NIST 2008 speaker recognition evaluation (SRE) data set [11]. All 8 conversation telephony data

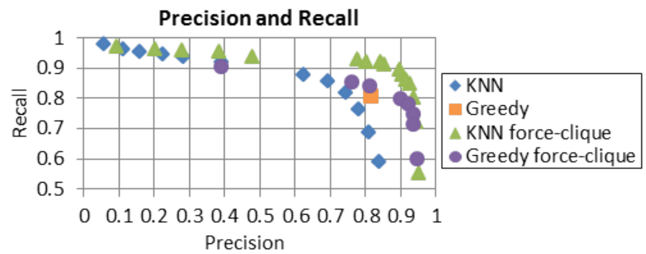

Fig. 1. Comparison of precision/recall for multiple algorithms on male NIST SRE08 data.

was used resulting in 2,790 male utterances and 4,748 female utterances. We measured performance in terms of edge precision and recall, degree distribution, and clustering coefficient. Whenever results from both genders aren't shown, we present results from the male subset; in all such cases results were comparable for the female corpus. In liu of performing cross validation on a different corpus, we compared results across genders and performed sensitivity analysis.

Edge Precision/Recall. The graph in Figure 1 shows the precision and recall of KNN for several values of k, greedy link prediction with the actual degree distribution, and force clique link prediction with several values of x applied to all of the above graphs. Force Clique link prediction improved precision and recall of both the graph generated by greedy link prediction and the k nearest neighbors graph. Even though the greedy algorithm had better performance than KNN, the KNN graph with force clique improvement was better than the greedy graph with force clique improvement. This is because many of the edges which were correctly generated by the greedy algorithm but not generated by the KNN could be generated by the force clique improvement algorithm.

Degree Distributions. Results are given in Figure 2. As expected, greedy link prediction generated a graph with almost perfect degree distribution. KNN had the worst degree distribution. Using $k = 8$ gave the closest distribution of all k values, but even this generated a degree distribution with a very different range and moderately different shape than that of the truth graph. Applying force clique-link prediction to the KNN graph made the generated degree distribution better match the truth degree distribution.

Clustering Coefficient. As seen in Figure 3, all of the methods produced graphs with reasonable clustering coefficients, indicating graphs where transitivity applies in a majority of cases. As when

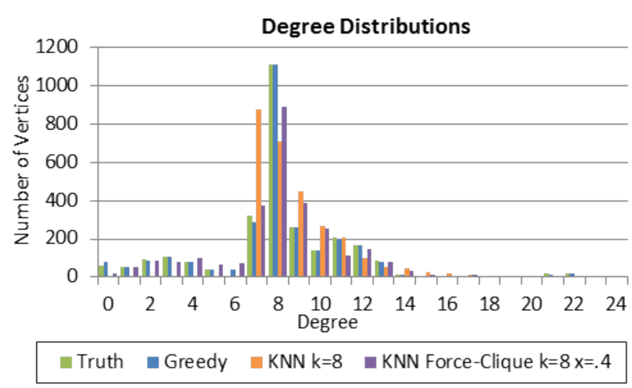

Fig. 2. Comparison of degree distributions on male NIST SRE08 data.

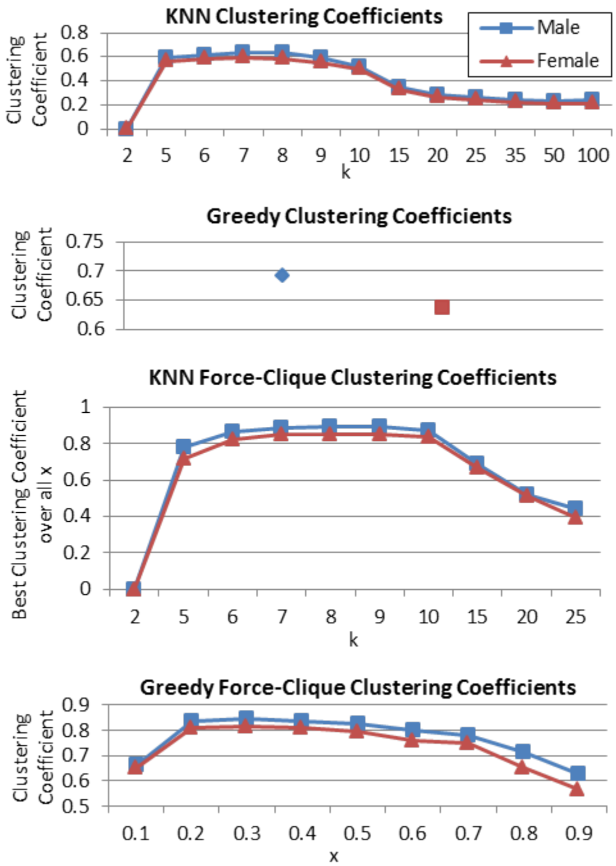

Fig. 3. Clustering coefficients for multiple algorithms.

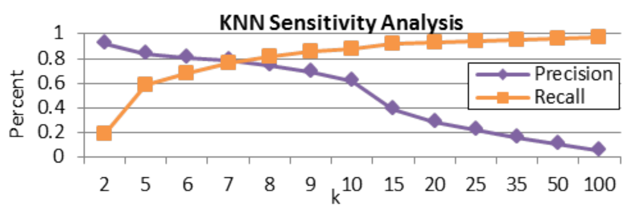

Fig. 4. Sensitivity analysis of the KNN algorithm.

measuring precision and recall, the best results were obtained by applying force-clique link prediction to a KNN graph.

5. PARAMETER SENSITIVITY ANALYSIS

Since it is highly unlikely that the optimal parameters will be known a priori, it is important to know how sensitive graph generation is to input parameters. Due to space constraints, we only consider the parameter sensitivity as it affects edge precision and recall on the male NIST SRE08 data.

KNN Sensitivity. KNN was the most sensitive of the algorithms we tested, see Figure 4. Precision seems to be more sensitive that recall, but once you deviate by more than one from the optimal value, both show significant changes.

Greedy Link Prediction Sensitivity. Greedy link prediction takes an entire distribution as input rather than just a single number, which makes sensitivity analysis especially important. The actual degree distribution of the entire data set may not be known exactly, but in many cases, the properties of the distribution may be known. Thus, it is reasonable to extract a close approximation to the complete degree distribution. In other scenarios, very little may be known about

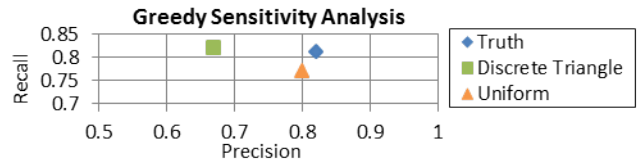

Fig. 5. Sensitivity analysis of the greedy algorithm.

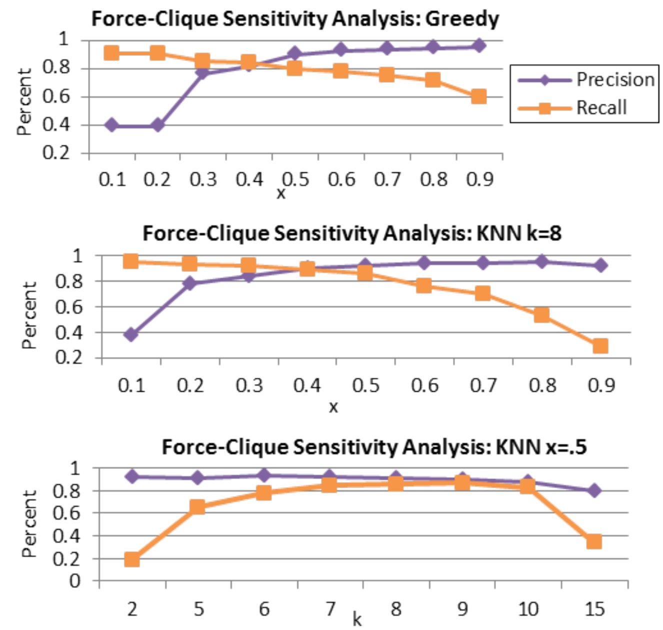

Fig. 6. Sensitivity analysis of the force-clique algorithm and interaction with other parameters.

the degree distribution. For comparison purposes, we consider two distributions which require very little a priori knowledge. In the uniform degree distribution, all vertices are given the same degree, which should be chosen to be the expected mean, $P(X = k) = 1$, $k =$ mean, and zero otherwise. In the triangle distribution,

$$P(X = k) = \begin{cases} \dfrac{2(k - \min)}{(\max - \min + 2)(\text{mean} - \min + 1)} & k \leq \text{mean} \\ \dfrac{2(\max - k)}{(\max - \min + 2)(\max - \text{mean} + 1)} & \text{otherwise.} \end{cases} \tag{6}$$

Results are shown in Figure 5. Using the uniform distribution, which only requires knowledge of the expected mean, yielded results almost as good as when the true distribution was used.

Force-clique sensitivity. Force-clique link prediction was somewhat sensitive to the choice of x, but setting $x = .4$ improved both precision and recall in all of our experiments and results were fairly robust near that point, see Figure 6. Additionally, applying force-clique link prediction reduced the senstivity of k in KNN.

6. CONCLUSIONS

Prior work evaluated the results of speaker recognition tasks [3, 5], but did not consider the quality of the content graph or the link prediction method in detail. We identified a set of metrics by which to judge speaker content graphs independently of the analysis tasks performed on them. We presented several methods for generating content graphs which outperformed the existing KNN method in all of those metrics. Future research will focus on using these improved graphs for speaker recognition.

7. REFERENCES

[1] W. Campbell and Z. Karam, "Simple and efficient speaker comparison using approximate KL divergence," in *Proceedings of Interspeech*, 2010.

[2] N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proceedings of Interspeech*, 2009.

[3] Z. Karam and W. M. Campbell, "Graph embedding for speaker recognition," in *Proc. Interspeech*, 2010, pp. 2742–2745.

[4] Z. Karam, W. M. Campbell, and N. Dehak, "Graph relational features for speaker recognition and mining," in *IEEE Statistical Signal Processing Workshop*, 2011, pp. 525–528.

[5] W. Campbell and E. Singer, "Query-by-example using speaker content graphs," in *Proceedings of Interspeech*, 2012.

[6] Mikhail Belkin and Partha Niyogi, "Using manifold stucture for partially labeled classification," in *Advances in Neural Information Processing Systems 15*, S. Thrun S. Becker and K. Obermayer, Eds., pp. 929–936. MIT Press, Cambridge, MA, 2003.

[7] Xiaojin Zhu, "Semi-supervised learning literature survey," Tech. Rep., University of Wisconsin–Madison, 2007.

[8] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *International Conference on Information and Knowledge Management (CIKM)*, 2003, pp. 556–559.

[9] D. A. van Leeuwen, "Speaker linking in large data sets," in *IEEE Proc. Odyssey Speaker and Language Recognition*, 2010.

[10] D. J. Watts and Steven Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[11] "The NIST year 2008 speaker recognition evaluation plan," http://www.itl.nist.gov/iad/mig/tests/sre/2008/, 2008.