RECOG: Recognition and Exploration of Content Graphs

Kara Greenfield∗ † ‡ Stephen Mayhew† ‡ William Campbell‡

MIT Lincoln Laboratory, Human Language Technology Group

Abstract
We present RECOG (Recognition and Exploration of COntent Graphs), a system for visualizing and interacting with speaker content graphs constructed from large data sets of speech recordings. In a speaker content graph, nodes represent speech signals and edges represent speaker similarity. First, we describe a layout algorithm that optimizes content graphs for ease of navigability. We then present an interactive tool set that allows an end user to find and explore interesting occurrences in the corpus. We also present a tool set that allows a researcher to visualize the shortcomings of current content graph generation algorithms. RECOG’s layout and toolsets were implemented as Gephi plugins [1].

Index Terms: H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing—Methodologies and techniques, I.2.7 [Artificial Intelligence]: Natural Language Processing

1 Introduction
Speaker Recognition is an umbrella term for the set of tasks pertaining to identifying a person by their voice and is an important forensics tool. Unfortunately, speech recordings by the same person contain inherent variability due to the effects of different recording environments (channels) and alterations of vocal characteristics by the speaker [2]. In spite of these difficulties, it is still possible to perform well on speaker recognition tasks by using speaker content graphs to look at the corpus as a whole [3].

Constructing a speaker content proceeds follows. Each node \( v \) in the graph corresponds to a single vector \( m \) from a speech signal. Letting \( d(m_i, m_j) \) denote the distance between signals \( m_i \) and \( m_j \), we define weights for edges in the graph via a weight matrix \( W = [W_{i,j}] \) where

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

The parameter \( \sigma \) 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 [4] for the distance in (1). In order for the graph to reflect the local neighborhood of \( v \), \( v \) is only connected to a limited number of other nodes. We use the algorithms described in [5] to determine which edges to include.

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

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

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

2 Graph Layout
RECOG’s layout algorithm optimizes content graphs for ease of navigability. First, we cluster utterances by speaker; this is done precisely when truth is known a priori or approximately by using a graph clustering algorithm [6]. RECOG creates a block model representation of the graph with a super-node for each cluster and a super-edge between two super-nodes if any of their constituent nodes were adjacent; the weight of the super-edge is the number of such adjacencies. The block model representation of the truth graph contains a super-node for each speaker and no super-edges. By convention, RECOG produces straight line drawings with spacing between super-vertices. Within the confines of these conventions, RECOG attempts to minimize edge crossings and edge length and to produce a drawing with good symmetry. Initially, the super-nodes are placed on cartesian grid points; the grid drawing is then mapped to a circular form in order to increase the angular resolution of edges. Finally, the constituent nodes and edges replace the super-nodes and super-edges.

The super-edges are sorted in decreasing order by weight and ProcessEdge is then applied to each edge. When super-node \( v_2 \) is added relative to super-node \( v_1 \), we consider a spiral originating at \( v_1 \) and place \( v_2 \) in \( v_1 \)’s subgraph in the first unoccupied position on the spiral.

Algorithm 1 ProcessEdge

\begin{align*}
\text{Require: } & \text{super-edge } e \text{ with incident nodes } v_1 \text{ and } v_2 \\
\text{Require: } & \text{optional relative position node } v_{rel} \\
& \text{if neither } v_1 \text{ nor } v_2 \text{ is in a subgraph then} \\
& \quad \text{if } v_{rel} \text{ was specified then} \\
& \quad \quad \text{Add } v_1 \text{ relative to } v_{rel} \text{ and add } v_2 \text{ relative to } v_1 \\
& \quad \text{else} \\
& \quad \quad \text{Create a new subgraph that contains } v_1, v_2, \text{ and } e \\
& \text{else if exactly one of } v_1, v_2 \text{ is in a subgraph then} \\
& \quad \text{Without loss of generality, let } v_1 \text{ be in a subgraph} \\
& \quad \text{Add } v_2 \text{ relative to } v_1 \\
& \text{else if } v_1 \text{ and } v_2 \text{ are in the same subgraph then} \\
& \quad \text{Add } e \text{ to that subgraph} \\
& \text{else if } v_1 \text{ and } v_2 \text{ are in different subgraphs then} \\
& \quad \text{Merge}(v_1, v_2)
\end{align*}

Algorithm 2 Merge

\begin{align*}
\text{Require: } & \text{Nodes } v_1 \text{ and } v_2, \text{ which are in different subgraphs} \\
& \text{Without loss of generality, let } v_1 \text{’s be larger than } v_2 \text{’s subgraph} \\
& \text{Sort the super-edges in } v_2 \text{’s subgraph in decreasing order by weight} \\
& \text{Store this sorted list in } edges \\
& \text{Remove } v_2 \text{’s subgraph} \\
& \text{Add } v_2 \text{ relative to } v_1 \\
& \text{for each super-edge, } e, \text{ in } edges \text{ do} \\
& \quad \text{ProcessEdge}(e, v_2)
\end{align*}

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.
The resulting layout is easily traversed by both users and developers and clearly displays anomalies in the data.

3 User Interactions

One of the main goals of RECOG is to enable an end user to explore the data. In addition to providing an easily navigable layout, we present several data analysis tools. The most important of these is a connection to the database that allows the user to inspect the audio, image, or video file associated with a node without needing to switch to an external program.

RECOG also contains more advanced tools for performing speaker recognition tasks. In the verification tool, the user can select a test utterance and a set of utterances comprising a target speaker and use the verification tool to determine the likelihood that the test utterance belongs to the target speaker. In addition to providing a raw numeric answer, the tool highlights the nodes and edges that affected the likelihood to enable the analyst to understand how that particular likelihood was computed so that they can manually adjust likelihood based on their subject matter expertise. There is a similar tool for performing speaker identification (selecting the best match from a set of target speakers).

In query by example (QBE), we are given a set of query utterances and must locate speakers that match the query set. RECOG visualizes the random walk approach developed in [3], providing the user with insight into the result.

4 Researcher Interactions

RECOG provides a content graph researcher with many tools to visualize why and how content graph link prediction algorithms achieve their results, enabling the researcher to improve the algorithms. In addition to presenting the standard precision and recall metrics, RECOG’s graph comparison tool colors correctly generated edges green, incorrectly generated edges red, and missed edges black, allowing the researcher to visualize how precision and recall were calculated. They can then use the database connection tool to listen to the recordings of any interesting nodes. RECOG also includes a dynamic visualization of generated content graphs, to allow a researcher to see not just which edges were included, but also the order in which they were generated; this is particularly useful when developing algorithms that generate edges with higher likelihood of being correct earlier in the process. The combination of these tools allowed us to develop link prediction algorithms with substantially better performance than the previous gold standard [5]. RECOG also has the ability to color nodes by a category different from that used to generate the the supernodes; this tool is especially valuable when developing graph clustering algorithms.

5 Conclusions

We presented RECOG, an interactive visualization system for speaker content graphs. It provides a navigable graph layout and tool sets for both researchers and analysts. Future work will involve user studies to measure the subjective qualities of these tool sets.

References