Ancient World Computational Analysis

AWCA is made possible through the Data Science Discovery Program at UC Berkeley, and the following people:

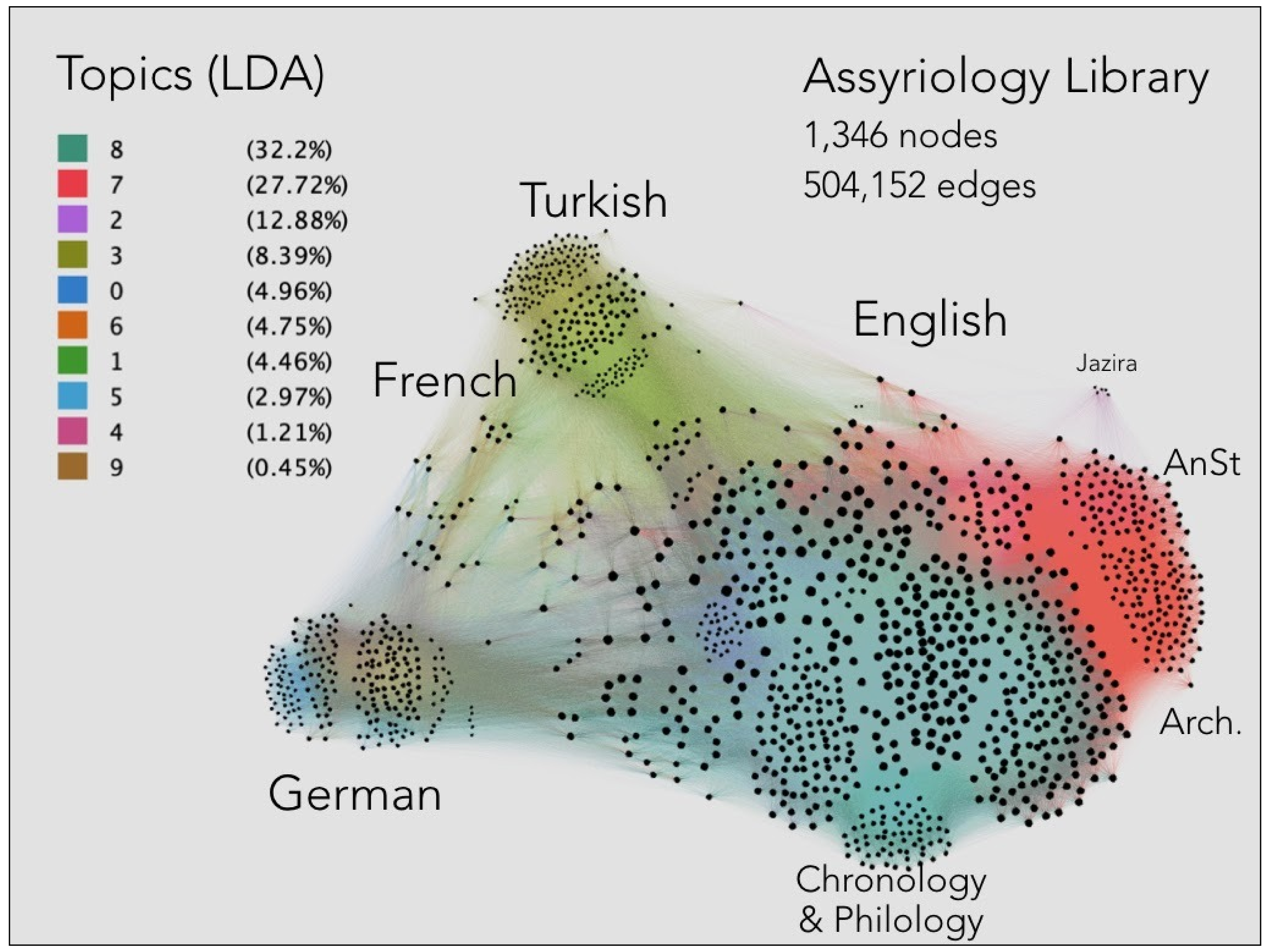
- **2020 team: Adam Anderson, Jason Webb, Kenan Jiang, Shrey Bhate.**
- **2019 team: Adam Anderson, Alan Cha, Jason Webb, Kenan Jiang.**
- **2019 visiting scholars: Aleksi Sahala, Ilya Akdemir, Isaac Dalke**

Project description

The goal is to visualize the research landscape of a body of texts from any collection of PDFs. The project utilized a collection which digitized books and journals from the field of ancient Near Eastern Studies, Classics, Archaeology, and Middle Eastern Languages. The result of this project applies different language models to the dataset to an edge list for network visualization and analysis.

These notebooks include the following tools:

1. **Optical Character Recognition (OCR)**: the Tesseract OCR in a JupyterNotebook. To use this the PDF must be converted to .TIF images. Coding credit goes to Alan Cha.
2. **Preprocessing Notebooks (NLTK)**: Two preprocessing Jupyter notebooks use the NLTK (natural languages tool kit) to process the OCR output as .TXT and .CSV files. These results will then be used for both citation analysis and bibliographic analysis. Coding credit to Kenan Jiang and Jason Webb.
3. **Topic modeling (LDA)**: the LDA results are assigned as weights for directed edges, with each source as the highest scoring documents within a topic, to each target drawn from the subsequent highest scoring documents within the same topic. The number of topics can be adjusted within the notebook, as can the number of edges drawn for each topic. Coding credit goes to Kenan Jiang.
4. **Doc2Vec & Word2Vec**: The results of the language model produce cosine similarity scores for any two documents in the dataset. The highest similarity scores (>.9) are assigned as weights in an undirected edge list. Coding credit goes to Kenan Jiang and Jason Webb.
5. **BERT** implementation. Coding credit to Kenan Jiang.



Tools for Citation & Content Analysis

This series of Python Jupyter Notebooks implements different types of NLP tools for computational text analysis, which is demonstrated on a large text corpus of multilingual, primary and secondary sources in the field of Near Eastern Studies. The methods and tools are intended to be generally applicable to any collection of documents (in many languages).

Results can be used to visualize the different types of language models in a network, thereby mapping the contours of the research landscape described within a collection of scholarly works. For the purpose of our initial proof of concept, we utilized a corpus of articles and monographs pertaining to the Old Assyrian Period (1950-1750 B.C.), primarily because it falls within Anderson's domain expertise (e.g., see Klaas R. Veenhof, Jesper Eidem, Mesopotamia. The Old Assyrian Period (OBO 160/5), Fribourg, Göttingen: Academic Press / Vandenhoeck & Ruprecht, 2008. 383 pp. + 2 maps. - ISBN 9783525534526).

Because the dataset includes primary and secondary sources still in copyright, the full text files are included only as 'bags of words' and 'bags of segments' for BERT. Copies of these works were obtained by scanning and OCR, which introduced numerous errors, especially concerning the transliterated cuneiform text. I hope to address these OCR errors downstream with deep learning models.

Network Visualization

The Gephi network files (.gexf) illustrate the results of the project in multi-modal networks of authors-to-authors (i.e. who cites whom?) and authors-to-primary sources (i.e. who cites what?), with links to the titles of the files, not the content (to avoid any copyright issues).

Jupyter notebooks with Google Colaboratory settings can be found in our GitHub with the folling links:

1) Tesseract OCR

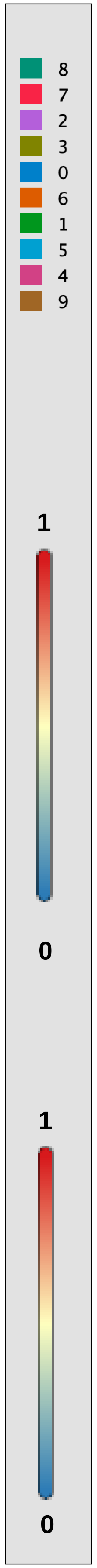
2) Preprocessing Notebook

3) TopicModel (LDA) Network

4) Doc2Vec Network

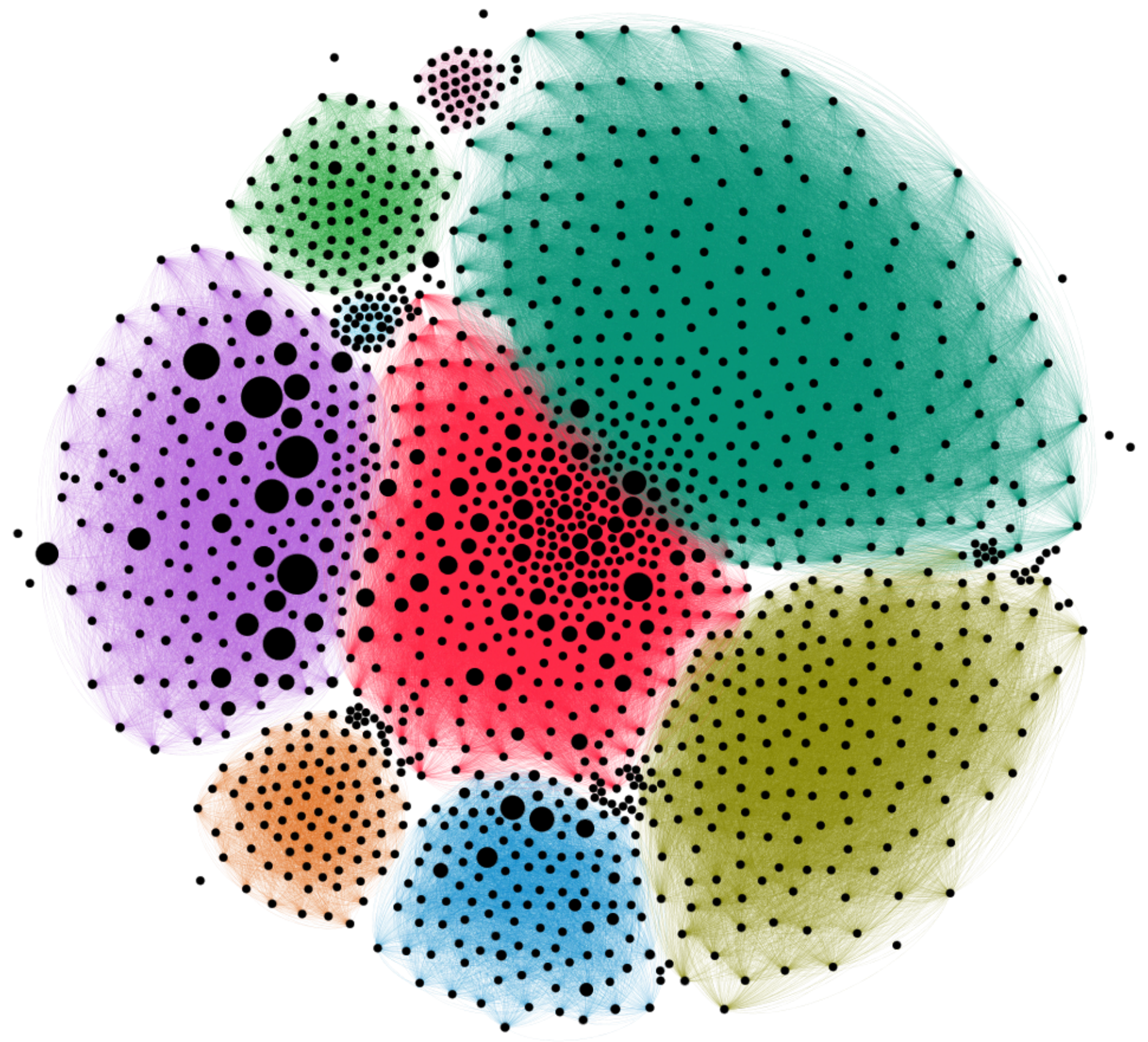
5) Word2Vec Network

6) BERT Network



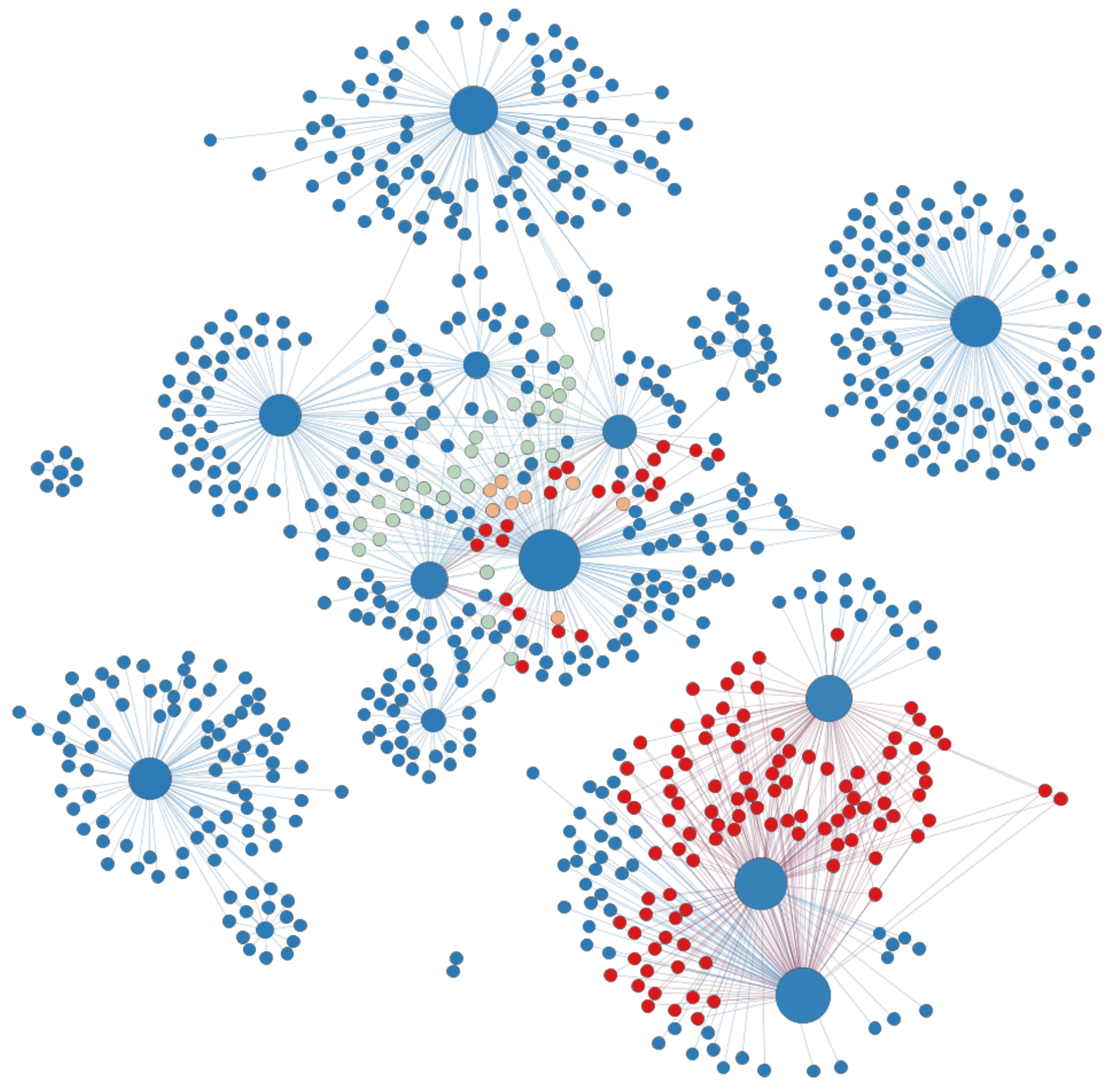
Topic Modeling (LDA)

1346 nodes / 116,028 edges
node size = betweenness centrality



Word Embedding (D2V)

769 nodes / 1,099 edges
node size = betweenness centrality
node color = clustering coefficient (0, 1)



Deep Learning (BERT)

149 nodes / 4,209 edges
node size = beteweeness centrality
node color = clustering coefficient (0, 1)