

# Math 525 Project Report

Connor McBride

Webster's Unabridged English Dictionary

## 1 Sourcing the Data

The source of the data comes from the 2009 English *Webster's Unabridged Dictionary* made available as an eBook by Project Gutenberg [Var09]. I used a repository by Matthew Reagan [Rea16] to parse the eBook file and turn the dictionary into JSON format. Downloading the JSON file, I was then able to write a Python script [McB25] that would convert the data into a NetworkX [HSS08] directed graph for analysis.

To generate the network, we first went through dictionary and created a node for each word that has a definition in the dictionary. We then iterated through each definition and created a directed edge from the word being defined to each other word appearing in its definition if that word in the definition is also defined. The weights for the edges are the number of times that the word appears in the definition (See Table 1.1 for example).

- Example:
  - "disgraduate" : "To degrade; to reduce in rank. [Obs.] Tyndale."
  - "disgraduate" -> "to" (2)
  - "disgraduate" -> "degrade" (1)

Figure 1.1: An example of edge creation. Five weighted edges are created in total, Two weighted edges are created from “disgraduate” to “to” and “degrade” with edge weight corresponding to the number of times it appears in the definition. “Obs” and “Tyndale” are not defined and therefore no edges are created to these words.

## 2 Network Analysis

### 2.1 Basic Structure

The basic structure of the dictionary network is given in Table 1. Certain definitions contained examples sentences using the word in them, meaning that we had 24,872 self-loops in the network. For certain calculations such as the k-cores, we had to remove the self-loops as the algorithm wouldn't work with them.

Number of Nodes	Number of Edges	Edge Type
102,217	1,704,423	Directed; weighted

Table 1: Basic Properties of dictionary network.

## 2.2 Centralities

Computing the centrality measures for the dictionary network presented some computational challenges due to the large number of nodes and edges. To deal with these issues, we used a combination of sparse array operations in SciPy [VGO<sup>+</sup>20] with the adjacency matrix and the igraph package [CN06] which implements the centrality algorithms in C.

The eigenvector and Katz centralities could be interpreted as “process” words that are useful when defining words. Betweenness centrality could be interpreted as the most common vocabulary words used. In-degree can be interpreted as the most utilized words in definitions while out-degree can be seen as the longest/most verbose definitions. In particular, “run” and “set” have the most definitions in the dictionary, which could help explain their high centrality measures. The top 4 words for each centrality can be found in Figure 2.1.

(a) Eigenvector		(b) Betweenness		(c) Katz	
Word	Centrality	Word	Centrality	Word	Centrality
set	0.0004389	a	0.02968116	set	1.7745e-05
run	0.0004184	of	0.02508660	run	1.6958e-05
take	0.0004115	to	0.02088582	turn	1.6621e-05
turn	0.0003981	see	0.01153032	take	1.6354e-05

(d) In-degree		(e) Out-degree	
Word	Degree	Word	Degree
a	59,405	set	750
of	57,317	run	674
the	52,744	turn	645
or	45,817	take	617

Figure 2.1: Centrality and degree statistics for dictionary network.

## 2.3 Connected Components

Examining the connected components, we see there are 573 weak components and 73,039 strong components. As demonstrated in Figure 2.2, the majority of components are quite small with just one large component. The largest strong component contains 27,637 words, while the largest weak component contains 101,642 nodes. This means that 99.4% of words are weakly reachable to each other. The small strong components are usually made up of different words that are part of the same word family, e.g. “lawyerlike” and “lawyerly.”

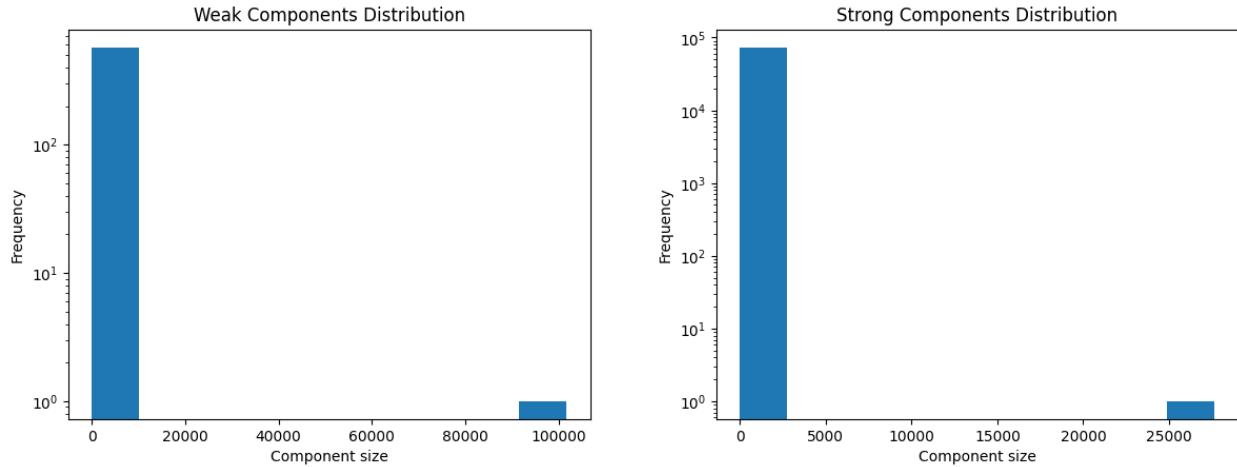


Figure 2.2: Plotted distributions of sizes of weak and strong components.

## 2.4 Community Detection

We employed the Leiden Algorithm written in C++ by the leidenalg package [TWvE19] for directed modularity groups. There are 621 detected communities with the largest one of size 33,285. The largest communities contained very common words while the smallest communities were often made up of word families. The most interesting communities that contained words making up specific scientific disciplines or other niche subjects. Three of which are shown in Figure 2.4.

We then build a meta graph, which takes uses detected communities as nodes and inter-community edges as edges. Examining the active meta-graph of the Leiden communities (nodes with inter-community edges), we can see that four communities are very influential, with edges connecting with 80-96% of the other communities.

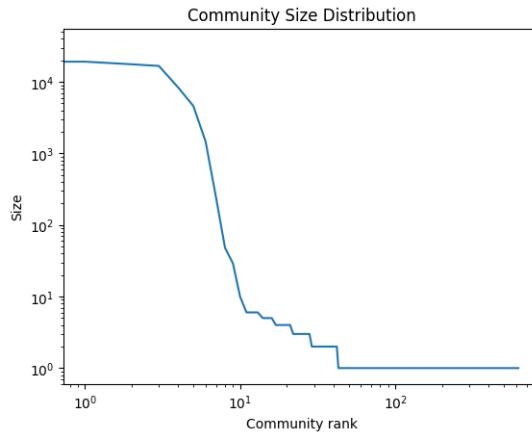


Figure 2.3: Sizes of detected communities.

(a) Seismic Community	(b) Pre-historic Community	(c) Infernal Community
Earthdin	Paleolithic	Satanical
Anaseismic	Neanderthal	Erebus
Terremote	Supraciliary	Devilish
Earthquake	Dolichocephaly	Infernal
Earthshock	Superciliary	Hellish
Earthquake		Diabolical
		Fiendlike

Figure 2.4: Three semantic communities computed along with appropriate nicknames.

## 2.5 K-Core Analysis

We next examine the  $k$ -cores of the dictionary network. The largest  $k$ -core was of degree 121 with 521 words. The largest  $k$ -cores were surprisingly large, as the cores of size 110+ contained thousands of words. Sampling example words from the different sizes of  $k$ -cores reveals that the cores of size 1 – 50 contain the more obscure words, while words in cores of size 120+ contain those that make up the semantic backbone of the dictionary (Table 2).

Core Size	Examples
1-50	anopheles, sarong, turcoman, corrugator
120+	I, most, some rest

Table 2: Example words for different  $k$ -core sizes.

## 3 Real World Properties

### 3.1 Power Law

The degree histogram resembles the power law pattern. The fitted value power law exponent  $\alpha \approx 1.452$ , meaning that it does not decay quite as fast as most real-world networks. This was done with `k_min = 1` and a calculated standard error of  $\sim 0.0013$ .

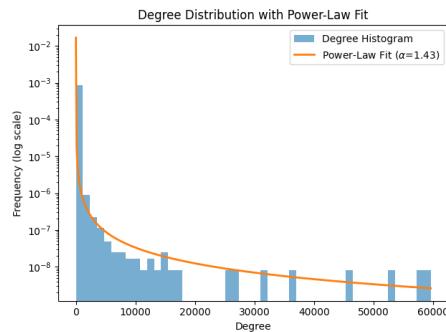


Figure 3.1: Power law curve fit to degrees distribution.

## 3.2 Network Measures

The network's degree assortativity of  $\sim -0.21$  means that the network is moderately disassortative, resembling more of a technological/biological network rather than a social network.  $\sim 0.042$  is very low reciprocity for a real-world network, but for a dictionary makes sense. Dictionaries try to avoid immediately circular definitions, but are not always perfect. Finally, the global clustering coefficient is  $\sim 0.215$ , and there exists evidence of strong local clustering. This would make sense as synonyms and words dealing with the same subject would tend to cluster. This value is typical for social and biological networks. Overall, these networks are consistent with numbers found in real-world networks.

Measure	Value
Degree Assortativity Coefficient	$\sim -0.210191$
Reciprocity	$\sim 0.042064$
Global Clustering Coefficient	$\sim 0.215269$

Table 3: Different network measures for the dictionary network.

## 3.3 Giant Component and Potential Models

As previously discussed, there exists a giant component in the weak sense, as 99.4% of the words belong to the giant weak component. There is not a giant component in the strong sense as only 27.04% of nodes belong the largest strong component. The existence of the weak giant component helps to argue that the network contains features of real world networks.

It is difficult to find a model that would incorporate all of the features characteristic of the dictionary network. One potential choice of model is a directed configuration model. This would allow us to specify that exact in-/out-degree sequences that we would expect and thus similar degree distributions and degree-based statistics. This model lacks, however, the desired clustering, reciprocity, and community structures that we have demonstrated.

## References

- [CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal Complex Systems*, 1695:1–9, 2006.
- [HSS08] Aric Hagberg, Daniel Schult, and Pieter Swart. Exploring network structure, dynamics, and function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy)*, pages 11–15, 2008.
- [McB25] Connor McBride. dictionary-network: A github repository. <https://github.com/connor-mcbride/dictionary-network>, 2025.
- [Rea16] Matthew Reagan. Webstersenglishdictionary: Webster’s unabridged english dictionary in json format. <https://github.com/matthewreagan/WebstersEnglishDictionary>, 2016.
- [TWvE19] Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.
- [Var09] Various. *Webster’s Unabridged Dictionary*. Project Gutenberg, 2009. eBook #29765, accessed <https://www.gutenberg.org/ebooks/29765>.
- [VGO<sup>+</sup>20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, Eric A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.