COMPARING LANGUAGE USE AND NETWORK STRUCTURE USING TWITTER*

CAITLIN SHENER[†], BRIAN OCEGUERA[‡], AND SEUNGJOO (SALLY) LEE[‡]

Abstract. We develop an approach for exploring questions regarding language use and connections between people in social networks. In particular, we investigate community structure and language usage in the network composed of the most followed ninety-nine users in Twitter. Our goal is to measure the relation between a community of users and the words employed by those users. We accomplish the investigation using k-means clustering to group users based on word choice, and we use modularity maximization and InfoMap clustering to find communities based on network connections. Our study illustrates how to mathematically analyze and interpret language use within social network structure.

Key words. social network | twitter | community detection | document clustering

AMS subject classifications. 94A15, 91D30, 90C35

1. Introduction.

Network scientists often question the extent to which similarity plays a role in forming social ties. They ask questions like, "Are similar people more likely to think, act, and communicate the same way as their friends?" Or, "Are we, as humans, drawn to form connections with people dissimilar from ourselves?" We use the term similarity here to refer to common interests and therefore common topics of conversation. The idea of "homophily" is defined as the principle that similar people will be more connected to each other than dissimilar people [11]. In order to easily determine if people who are more connected discuss the same topics, we develop an approach to determine how homophilic word use functions in networks.

Our method of comparing word use to social ties involves creating groups in two distinct ways, and then measuring the overlap of these groups:

1. We created **Network Communities** through modularity maximization and Infomap; such an approach uses information from data on the edges in the network. We explain these methods in further detail in section 3.
2. We also created **Language Clusters** through k-means clustering; such an approach uses information from data on the words used in the tweets.

Although some have considered language use in their analysis of networks [2], our approach uses distinct groupings based on the term-frequency inverse-document-frequency model in section 4 of this paper. This allows researchers to avoid asking the question: "How different is the language use between network communities?" after creating the communities. Instead, researchers can ask, "How much do language clusters overlap with network communities?" The approach allows for the consideration of language use along with network structure in community detection, which is important since algorithms for community detection on networks are still contested

*Submitted to the editors November 5, 2018. Completed under the guidance of Mason A. Porter, Professor in the Department of Mathematics, University of California, Los Angeles (mason@math.ucla.edu).

[†]Department of Statistics, University of California, Los Angeles, CA, 90095 (caitlinshener13@ucla.edu).

[‡]Department of Mathematics, University of California, CA 90095 (brianoceguera@ucla.edu, sally.lee1725@gmail.com).

today [4]. Additionally, our approach incorporates a consideration of language use in finding clusters in the network that can be applied to many different types of networks. The last benefit to our approach is the ability to produce comparable and interpretable values when comparing partitions on the network. The z-scores that we calculate in section 5 allow us to answer how much language use overlaps with the social ties in a given network.

To illustrate our approach, we use the social media site Twitter. Twitter has become a popular platform for users to connect to other users, including people who they have never met. A Twitter user can "follow" another user, which allows the former to see recent updates from the latter in the form of "tweets". A tweet consists of a maximum of 140 characters[1] and, for example, can be a short text snippet describing a user's current opinions or business promotions. For many users, Twitter is a stage for expression and identity creation, where language is an important currency for influencing others and spreading information.

We construct a network [14] of the top ninety-nine users followed on Twitter. We represent each user as a node, and create directed edges pointing from one user to the users they follow. For the Twitter follow network, one can intuitively consider the network as a social network, because users interact with others in whom they are interested and possibly know. However, Twitter differs from many other social networking sites, because many users often follow accounts of celebrities or news sources and tweet at users who are unaffiliated with them. Because of the duality behind Twitter's function as a social network and Twitter's usage for the dissemination of information, a Twitter network has characteristics of both a social network and an information network [12].

Through our language and network analysis of the top Twitter users, our approach considers whether users who write about similar topics are more likely to be in the same network community. Essentially, we ask: "Do people in the same communities, based on network structure, discuss similar topics?"

The rest of the paper is organized as follows. In section 2, we discuss how we gathered and represented the data from Twitter. In section 3, we discuss the methods of network community detection that we used to explore the network. In section 4, we explain how we processed the tweet corpus to cluster the network based on tweet language. In section 5, we detail some calculations that we will need for our analysis in the following section. In section 6, we discuss how to interpret our results. In section 7, we further explore and interpret our results and draw conclusions. We also touch on possible further studies based on our initial research.

2. Data Acquisition. To gather the network information from Twitter, we made use of Twitter's REST API [26]. The API has methods that allow one to make rate-limited queries to the company's database for information about users. Due to time constraints and Twitter's rate limitation (which throttled our requests to a limited selection every 15 minutes), we decided to investigate a small sample size (of one hundred Twitter users). For convenience and accessibility, we chose the one hundred most followed users on Twitter. However, we then excluded one Twitter user, @MohamedAlarefe, as he was both a language and a network outlier. As the only user in our dataset tweeting in Arabic, he did not follow or receive follows from any of the other ninety-nine users. Although the majority of users in our dataset

[1] At the time of our data gathering, tweets were limited to 140 characters. Recently, Twitter increased its character limit to 280 for tweets in English and other languages. For more information, please see https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html.

wrote in English, we also included a smaller subgroup tweeting in Spanish.

2.1. Creating a Directed Adjacency Matrix. Because Twitter does not have publicly available datasets regarding follower networks on their website, we built an unweighted, directed adjacency matrix A_{ij} ourselves using the results from Twitter API queries. We define A_{ij} as the directed adjacency matrix where each entry a_{ij} in the matrix is 1 if j follows i and 0 otherwise. Thus, A_{ij} is a matrix of ones and zeros. In the Twitter network, A_{ij} has dimensions 99×99. The process of creating the adjacency matrix involved taking the results from the Twitter queries (which return information as a JSON-formatted string) and parsing follower information as an edge list. In an edge list, there is an edge of weight 1 from user A to user B if B follows A. In the final representation of the network with the top ninety-nine users, we labeled nodes by Twitter user name.

2.2. Network Measures. We now define some of the concepts that we used to quantitatively measure network structure on our network of Twitter users. See [14] for further discussion.

In-degree/Out-degree. A degree of a vertex or node is the number of edges connected to that vertex. In directed networks, such as our dataset, the in-degree of a vertex represents the number of followers of a specific Twitter user. Likewise, the out-degree represents the number of users in the network that a specific Twitter user follows. The mean in-degree and out-degree over the set of all vertices in our Twitter network is 15.2755.

Path. A path is a sequence of vertices such that every consecutive pair of vertices in the sequence is connected by an edge in the network. A path is defined in both the directed and undirected case. In general, a path can intersect itself by revisiting a vertex or traversing an edge or set of edges multiple times.

Shortest/geodesic path length. A shortest path between two vertices is the shortest possible distance (in terms of the number of edges) needed to traverse from one vertex to the other. The mean geodesic path length of our network is 2.088

Diameter. A network's diameter is the length of its longest geodesic path. The diameter of our network is 5.

Strongly connected component. A strongly connected component within a directed network is a set of vertices where there exists a bidirectional path between any two pairs of vertices. The size of the largest strongly connected component in our Twitter network is 78. Because the largest connected component is close to the size of the overall network, a significant portion of the network is interconnected through mutual following.

2.3. Downloading Tweet Data. Using a combination of Python libraries that serve as wrappers for Twitter's REST API method calls [22, 25], we downloaded and cleaned the latest 200 tweets from each of the top ninety-nine users. We chose the number 200 to successfully work within the constraints of Twitter's API. Using TWEEPY [25], we queried Twitter to return the last 200 tweets from each of the top ninety-nine users. We will refer to the set of tweets from all ninety-nine users as the *tweet corpus*.

We downloaded the data on June 5th, 2017. We expected a total number of $99 \times 200 = 19,800$ tweets, but instead the total number of tweets in the tweet corpus is 19,579. The reason for the smaller tweet count is that there were a small number of users with an entire tweet history fewer than 200 tweets. We decided to ignore this inconsistency in our textual analysis of the tweets. For reference, the dates of the tweets range from June 3rd, 2011 to June 5th, 2017. When inspecting the dataset, we found a single user (@Adele) with infrequent tweets (mostly dating from 2011 to 2014). If we chose to exclude Adele from our set of users, the oldest user's tweet occurs on July 13th, 2014. Interestingly, the median tweet timestamp occurs on April 25th, 2017, which indicates that most of these influential users tweet very often. The mean timestamp occurs on February 7th, 2017.

3. Community Detection in the Network. [4, 16] A community is a group of vertices which are densely connected by edges inside the group and sparsely (less densely) connected to vertices outside the group. The distinction between dense and sparse depends on the resolution parameter. A larger resolution parameter leads to a smaller number of total communities. There are several algorithms for determining communities in a network, and different community detection algorithms may determine different communities inside a given network. As such, when referring to "the community of node i" or c_i, we mean the community to which i belongs according to the community finding algorithm under discussion. Some formulations allow communities to overlap. In our formulation, we do not allow communities to overlap.

We used two methods of community detection in the network. Both are based on random walks on a network [10]. The first method we used was modularity maximization, which seeks to partition a network into clusters such that there are more edges between nodes of the same cluster than between those of different clusters [4]. In terms of random walks, the method of modularity maximization tries to maximize the length of a random walk contained in the same community [10]. We use the traditional modularity function [13] with various values of the resolution parameter to achieve different numbers of communities (see Table 2). With a resolution parameter of 1, the network partitioned into six communities, but we used values above and below 1 to get partitions between four and eight communities; a larger resolution parameter leads to a smaller number of communities.

Because our network is directed, we must modify the undirected form of modularity into the directed form. Below is the undirected form of modularity,

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)\,,$$

where m is the total number of edges in the unweighted network, A_{ij} is the ij^{th} element of the adjacency matrix, k_i and k_j represent the degree of node i and node j respectively, and $\delta(c_i, c_j)$ is the Kronecker delta function (which is 1 node i and node j are in the same community and is 0 otherwise). The null model, $P_{ij} = \frac{k_i k_j}{2m-1} \to \frac{k_i k_j}{2m}$ in the limit of large m, represents the probability that node j is connected to node i according to the configuration model [9]. A graph in the configuration model is generated as follows: given a set of degrees where each degree is mapped to a node in the network (degree sequence), a node has edge "stubs" that are connected uniformly at random [14].

In transitioning to the directed case, P_{ij} must take into account the in-degree and out-degree of each node. As a result, P_{ij} becomes a directed version of the configuration model. This yields the following expression for modularity in the directed case [9]:

$$Q = \frac{1}{m}\sum_{i,j}\Big[A_{ij} - \frac{k_i^{\text{in}}k_j^{\text{out}}}{m}\Big]\delta(c_i, c_j)\,, \tag{1}$$

where k_i^{in} and k_j^{out} represent the in-degree of node i and out-degree of node j.

The algorithm that we used for calculating Q is based on minimizing the Hamiltonian $\mathcal{H}$. The Hamiltonian provided by [19] can be rewritten in our notation as

$$\mathcal{H} = -\sum_{i,j}(A_{ij} - \gamma P_{ij})\delta(c_i, c_j)$$

where γ is the resolution parameter. Our definition is the same as in [19], where the pair use σ_i rather than c_i to denote the spin state of a node i in a graph. As shown in [19], when setting the resolution parameter $\gamma = 1$, one quickly notices that $\mathcal{H}$ and Q are directly proportional by a constant $-\frac{1}{m}$. Consequently, modularity can be written as

$$Q = -\frac{1}{m}\mathcal{H}\,. \tag{2}$$

Since the Hamiltonian is negative, we can maximize modularity by minimizing $\mathcal{H}$. We used the Python library BCTPY [8] along with NETWORKX [5] to carry out the above calculations for directed modularity on the social network. Specifically, we used the method "modularity_dir" that uses a deterministic modularity maximization method and a resolution parameter of 1 to compare the clustering results to the word clustering results, which we discuss in section 4.2.

The second algorithm we used for partitioning is InfoMap, a method that uses random walks to determine community structure [2] to represent information flow in a network [21]. A typical random walk across a network can be represented by a string of code words, where each word in the string corresponds to a node that the random walk visited in sequential order. InfoMap uses communities to shorten this representation by using a two-layered coding system where each community has a separate code word associated with the random walk entering and leaving the community, and each community also has a code system for the nodes within the community that are specific to that community [10]. Consequently, nodes in different communities can have the same code word, but because of the coded entry and exit words of the communities, these nodes can be differentiated from each other in the random walk. One finds an optimal partition of a network by solving the problem of how to most concisely represent random walks across the network by changing the encoding method of these walks based on the partitions. With the network partitioned using this optimal partition, a random walker should remain in the same community for a long time before exiting to a different community [10]. We used the implementation of the InfoMap algorithm from the IGRAPH software library [3], in R [17], using a method called "cluster_infomap."

4. Clustering By Language.

[2]Community structure refers to the state of having grouped nodes in a graph according to their respective community. To "determine community structure," means the process of determining all of the communities in a graph (including which node belongs to which community) by using a community detection algorithm.

4.1. Preprocessing. While one can, in principle, download the entire tweet history of a set of Twitter users and compare language use across individual tweets, we instead seek to acquire a general sense of what each user discusses over many tweets rather than in an individual tweet. We seek to find a way to summarize a Twitter user's language usage by extracting their most-used words while filtering out irrelevant words (such as articles) that do not add useful information for study. After doing so, we can compare users through language usage. Therefore, we grouped the last 200 tweets of each Twitter user into one large text: the tweet corpus. We parsed the tweet corpus into a dictionary of key-value pairs, where each key is a user and each value is a string of that user's 200 tweets put together. We then utilized Python's NLTK library [1] to process the words used by each user. Before applying text-based clustering, the words must be filtered so that only relevant words remain. In following paragraphs, we discuss the text filtration process.

First, we use NLTK's built-in list of English *stop words*. Stop words are words that do not have any real meaning associated with them. In particular, stop words consist of articles, conjunctions, and prepositions that only serve to connect other, more important words together. Although NLTK's list of stop words only include English words, we decided not to include other languages because an overwhelming majority of the tweets were in English and we suspect that the use of other languages will play a role in connections.

After removing the stop words, we used NLTK's built-in word stemmers to stem each word [1]. The process of stemming words involves taking a word and condensing it into its word stem. For example, the words "walking" and "walks" both become the word stem "walk". To account for words occurring too frequently or too infrequently potentially skewing our results, we remove any words that appear in under twenty percent or over eighty percent of the users' tweets.

After preprocessing the list of tweets for each user, we used the NLTK library to build a term frequency-inverse document frequency matrix (tf-idf). The tf-idf matrix is a product of two statistical measures: term frequency and inverse document frequency. The term frequency statistic measures the number of times a word appears in a text, such as the collection of a user's tweets. The dimension of the term frequency matrix is $99 \times p$, where p is the number of stemmed words across the entirety of the tweet corpus. The inverse document frequency statistic weighs words by how often they occur across a set of documents, such as the entire tweet corpus from all ninety-nine users in our dataset. The dimension of the idf matrix is $p \times 1$. The idf statistic assigns less weight to very common words and more weight to unique words in the entire tweet corpus.

For our dataset, the term frequency matrix contains each user's tweet collection as a row header and each stemmed word as a column header. To fill in the elements of the matrix, we calculate the term frequency of each word with respect to each user's tweet collection. As a result, many entries of the matrix are zeros because it is rare when the same term is used by many users. We then multiply the term frequency by the idf matrix. The idf matrix is computed as follows:

$$\text{idf}(t) = \ln\left(\frac{n_d}{f_d(t)}\right), \tag{3}$$

where t is a given term from our tweet corpus and $n_d = 99$ is the total number of documents or tweet collections (one per user). Also, $f_d(t)$ is the frequency of documents or tweet collections that include the word t. Note $f_d(t) \geq 1$, because any given word t appears in at least one document.

In other words, the dimension of the tf-idf matrix is 99×1. We then normalize the tf-idf vector v by the Euclidean norm:

$$\hat{v} = v_{\text{norm}} = \frac{v}{\|v\|_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}}\,. \tag{4}$$

4.2. K-means Clustering. Given the tf-idf matrix, we seek to partition the data into distinct groups based on word usage. To obtain a partition, we use k-means clustering [7], which attempts to minimize:

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}, \tag{5}$$

where

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\,, \tag{6}$$

and C_k denotes the set of observations in the kth cluster. Equation (6) is the sum of all the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations in the kth cluster. Combining equations (5) and (6), the clustering procedure becomes an optimization problem:

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_K} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}. \tag{7}$$

To perform k-means clustering, we use the SCIKIT-LEARN library in Python [15]. One important thing to note about k-means clustering is that it requires specifying the desired number, k, of clusters. We decided to match k to the number of communities generated by the community-detection algorithms from section 3. We computed k-means with $k = 4, 5, 6, 7, 8$.

We present an example of 5-means clustering in Table 1. As one can see from Table 1, the algorithm was able to group users based on common tweet topics. For example, Cluster 3 is centered around people who recently talked about the NBA Finals. Cluster 2 is centered around current politics and the then-recent attack on the London Bridge. Table 1 shows that the use of k-means clustering yields clusters with words that have topical similarity, as opposed to clusters based on arbitrary word usage. We suspect that the temporal nature of conversation topics within Twitter play a role in our clustering. Also, recall that our tweet corpus consists of the last 200 tweets from each user. In future work, one could collect the entire tweet history of these users to see how tweets over a longer time period affect the clustering results.

Through through modularity-based community detection with a resolution of 0.8 we found 5 different communities. In Figure 1 we lable these communities with letters A through E. The size of each community from A to E is 23, 29, 1, 44, and 2. As seen in Figure 1, the clusters found by k-means clustering overlap slightly with the communities found through modularity-based community detection with a resolution of 0.8. For example, Community A is the only community with the pink cluster, and Community B is the only community with the orange cluster. However, both shades of green and purple are in multiple communities. We provide a detailed way to determine the overlap of clusters and communities in the next section.

TABLE 1
K-Means Clusters for $K = 5$

K-means	Number of Users	Most-Used Terms Per Cluster
Cluster 1	49	rt, new, love, thank, tonight, day
Cluster 2	6	attack, london, police, trump, terror, say
Cluster 3	5	nbafinals, game, espn, valverde, sportscenter, warriors
Cluster 4	31	love, u, thank, rt, tonight, happy
Cluster 5	7	en, el, que, la, para, por

Modularity Parameter 0.8 Algorithmic Community Detection and 5-Means Language Clustering

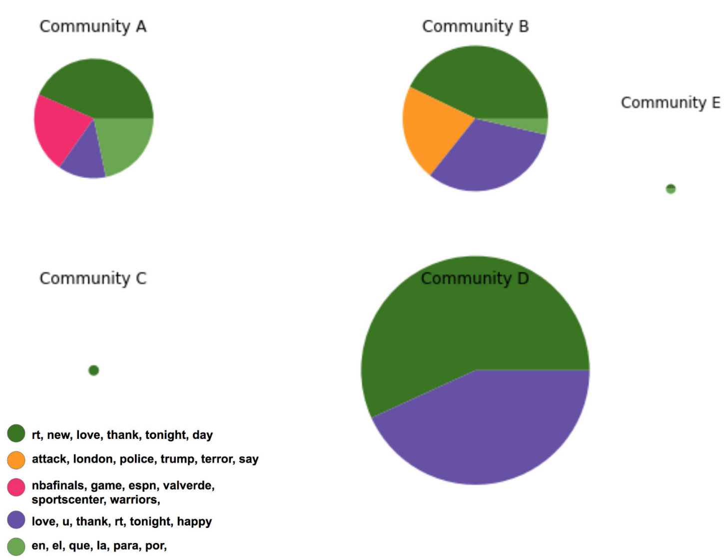

FIG. 1. *Each community, which we indicate with a letter, is the output of detecting communities by maximizing modularity as described in section 3 with a resolution parameter of .8. The size of the pies reflects the number of users in an algorithmic cluster. The color represents text-based clusters outlined in 1. This figure serves as a visual demonstration of the extent to which clusters determined by language usage and communities determined by network structure overlap.*

5. Z-Score Calculations. To answer our original question of whether connections between users in the network reflected similar content in their tweets, we looked at how the users grouped based on the structure of the network (so-called "network communities"), as opposed to the language that they used (so-called "language clusters"). To compare the language clusters and network communities, we calculated the Rand coefficients [18] of the combination of one network grouping and one language grouping, and then determined the z-scores of these coefficients [6, 23]. The Rand coefficient is a measure of the similarity of two partitions on the same data. It ranges from 0 to 1, where 0 indicates no similarity and 1 indicates the two partitions are identical. Although the distribution of the Rand coefficients is asymptotically Gaussian, and the z-scores and associated p-values of the coefficients cannot be interpreted as exact measures of significance, the z-scores and p-values are still good approximations of the probabilities of seeing the given outcomes.

In comparing the network communities against our language clusters, finding pairs of people is of great importance. Each node in our network is a person, and if node pairs are assigned to the same group in the network community and also the language cluster then our clustering methods have some similarity. In adopting the Rand coefficient to compare partitions, we define M to be the total number of pairs of nodes in our network, M_1 to be the number of pairs in the same network community, M_2 to be the number of pairs in the same language cluster, and w to be the number of pairs that are in the same network community and in the same language cluster. The Rand coefficient is the $S = [w + (M - M_1 - M_2 + w)]/M$ [18] and the z-score is

$$z = \frac{1}{\sigma_w}\Big(w - \frac{M_1 M_2}{M}\Big), \tag{8}$$

where

$$\begin{aligned}\sigma_w^2 &= \frac{M}{16} - \frac{(4M_1 - 2M)^2(4M_2 - 2M)^2}{256M^2} + \frac{C_1 C_2}{16n(n-1)(n-2)} \\ &\quad + \frac{[(4M_1 - 2M)^2 - 4C_1 - 4M][(4M_2 - 2M)^2 - 4C_2 - 4M]}{64n(n-1)(n-2)(n-3)},\end{aligned}$$

$n = 99$ is the total number of nodes in the network,

$$C_1 = n(n^2 - 3n - 2) - 8(n+1)M_1 + 4\sum_i n_{i.}^3,$$

and

$$C_2 = n(n^2 - 3n - 2) - 8(n+1)M_2 + 4\sum_j n_{.j}^3.$$

The summation terms in the calculation are based on a contingency table of network communities and language clusters, where the ij^{th} element of the table is the number of nodes in the i^{th} cluster based on network structure and the j^{th} cluster based on language use [23]. Here, $n_{i.} = \sum_j n_{ij}$ is the row sum of the contingency table, signifying the number of nodes in the ith network cluster. Similarly, $n_{.j} = \sum_i n_{ij}$ is the column sum of the contingency table, signifying the number of nodes in the j^{th} language cluster [23].

6. Results. As mentioned in [23], there are issues with using z-scores that are important to consider when interpreting results. Because the distribution of Rand coefficients is not Gaussian and often has heavy tails, it can be common to observe extreme z-score values. Nevertheless, even using an approximation of a Gaussian distribution is sufficient to claim significance for large enough z-scores. Although we ultimately do not claim significance, calculating z-scores is one viable way to compare our language clusters and structural communities.

Table 2 gives the z-scores between different comparisons of community divisions based on different methods. We compared the language clusters using various values of k to network communities derived using different methods (modularity at different resolutions and InfoMap clustering). As seen in Table 2, our methodology allows us to use a variety of k values and community-detection algorithms. From Table 2, we can also see that if we assume a Gaussian distribution, fourteen of the twenty-five

Table 2
Z-Scores for Comparison of Partitions

K-means	Mod (res=0.75)	Mod (res=0.80)	Mod (res=1.0)	Mod (res=1.055)	InfoMap
K = 4	**3.8115**	4.2733	1.3161	0.8090	1.4234
K = 5	3.9427	**3.7199**	1.5309	0.9298	1.9257
K = 6	3.9427	3.7199	**1.5309**	0.9298	1.9257
K = 7	7.7907	6.8025	4.7976	**3.0046**	0.0911
K = 8	12.4150	9.4211	5.7226	4.4967	**1.6520**

"K-means" represents the language clusters based on k-means clustering using the given value of k. "Mod" indicates partitioning using modularity maximization using the associated resolution. "InfoMap" indicates partitioning using InfoMap. The bold numbers across the diagonal signify comparisons of language-based partitions with network-based partitions with the same number of clusters.

z-scores would be statistically significant at a 1% level. Consequently, for the two specific methods of clustering that we consider, it is unlikely that the partitions of the network based on structure compared to partitions based on language would agree to the extent that we observed based on chance alone. Because our results did not exceed what previous papers have used as a threshold for significance [24] and the distribution of z-scores is not Gaussian, we only claim that our partitions are slightly similar, but not at a significant level. Our z-scores, however, reveal that comparing network structure to language can be compressed to a few numbers for easy analysis. For example, from the z-scores we can infer that while language use may play a role in how Twitter users create connections (i.e., follow others), language use is not a dominant factor in driving Twitter connections, and there are other factors that seem to be influencing social connections.

7. Conclusions and Discussion. Through our study we developed a new approach for answering questions about how people communicate and form connections with others.

In our example, we found that there is a slight relationship between language clusters and network communities in how similar they are among the top ninety-nine users on Twitter. Because we only used a small sample of users and only looked at their word usage in their tweets at a specific time, our conclusions only apply to this specific group of users within the time frame we examined them. Our results are influenced by our small sample size and our selective choice of subjects because many of the users in our dataset come from similar industries (specifically, the entertainment industry). Our dataset reflects a snapshot of popular culture at that time; many of the clustered words center around current events. To build on our work, we suggest that others examine larger datasets, also using other social networking sites with similar friendship structure, to see if there also exists similarities between network communities and clusters based on other characteristics, like language.

Our analysis of the top ninety-nine Twitter network illustrates that the words that users post can play a role in their connections on Twitter. As seen in Table 1, words that users of a certain network community use can center around certain topics, such as sports or news. Accordingly, what people talk about is fairly related to the people with whom they are connected. However, it is necessary to be cautious when interpreting our results. The z-values are not as large as in other papers [24]. While a user's language may play some role in community structure, it is not the only determining factor for connectivity between users.

Future studies also focusing on Twitter could adapt our approach to look at other network groups, either larger or smaller, and perhaps selected based on other criteria. Determining if the results are consistent across varying sizes and structures of networks might lead to more insight on what underlying factors contribute to network structure.

Additional studies could also explore larger and more diverse social and information networks (in addition to Twitter). Another example of a network that uses language would be the user network of Instagram users and the use of language in hashtags. Future studies could apply our approach to this network to see if similarities in language clusters and network communities appear in this network as well. In addition, the communities in a social network change over time, and the content depends heavily on news, especially groundbreaking pieces. Capturing the changes in communities both through language and network structure over time, and especially as they change with current events, would also be an interesting extension of our study.

Acknowledgments. We would like to thank the UCLA Mathematics Department as well as Professor Mason Porter for his guidance throughout our project. We would also like to acknowledge brandonrose.org [20] for their post on “Document Clustering with Python,” which provided guidance for the text clustering part of our work.

REFERENCES

[1] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, Inc., 2009.

[2] J. Bryden, S. Funk, and V. A. Jansen, *Word Usage Mirrors Community Structure in the Online Social Network Twitter*, EPJ Data Science, 2 (2013), p. 3.

[3] G. Csardi and T. Nepusz, *The IGRAPH Software Package for Complex Network Research*, InterJournal, Complex Systems (2006), p. 1695, http://igraph.org.

[4] S. Fortunato and D. Hric, *Community Detection in Networks: A User Guide*, Physics Reports, 659 (2016), pp. 1–44.

[5] A. A. Hagberg, D. A. Schult, and P. J. Swart, *Exploring Network Structure, Dynamics, and Function Using NetworkX*, in Proceedings of the 7th Python in Science Conference (SciPy2008), Pasadena, CA USA, Aug 2008, pp. 11–15.

[6] L. Hubert, *Nominal Scale Response Agreement as a Generalized Correlation*, British Journal of Mathematical and Statistical Psychology, 30 (1977), pp. 98–103.

[7] G. James, D. Witten, T. Hastie, et al., *An Introduction to Statistical Learning*, vol. 6, Springer, 2013.

[8] R. LaPlante, *Brain Connectivity Toolbox for Python (version 0.5.0)*, 2016, https://github.com/aestrivex/bctpy (accessed 2017-06-16).

[9] E. A. Leicht and M. E. J. Newman, *Community Structure in Directed Networks*, Physical Review Letters, 100 (2008).

[10] N. Masuda, M. A. Porter, and R. Lambiotte, *Random Walks and Diffusion on Networks*, Physics Reports, (2017).

[11] M. McPherson, L. Smith-Lovin, and J. M. Cook, *Birds of a Feather: Homophily in Social Networks*, Annual review of sociology, 27 (2001), pp. 415–444.

[12] S. A. Myers, A. Sharma, P. Gupta, and J. Lin, *Information Network or Social Network?: The Structure of the Twitter Follow Graph*, in Proceedings of the 23rd International Conference on World Wide Web, ACM, 2014, pp. 493–498.

[13] M. E. J. Newman, *Finding Community Structure in Networks Using the Eigenvectors of Matrices*, Phys. Rev. E, 74 (2006).

[14] M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, 2010.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, et al., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[16] M. A. Porter, J.-P. Onnela, and P. J. Mucha, *Communities in Networks*, Notices of the

AMS, 56 (2009), pp. 1082–1097.
[17] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, http://www.R-project.org. ISBN 3-900051-07-0.
[18] W. M. Rand, *Objective Criteria for the Evaluation of Clustering Methods*, Journal of the American Statistical Association, 66 (1971), pp. 846–850.
[19] J. Reichardt and S. Bornholdt, *Statistical Mechanics of Community Detection*, Physical Review E, 74 (2006).
[20] B. Rose, *Document Clustering with Python*, 2015, http://brandonrose.org/clustering (accessed 2017-06-15).
[21] M. Rosvall and C. T. Bergstrom, *Maps of Random Walks on Complex Networks Reveal Community Structure*, Proceedings of the National Academy of Sciences, 105 (2008), pp. 1118–1123.
[22] The Python-Twitter Developers, *A Python Wrapper Around the Twitter API*, 2017, http://python-twitter.readthedocs.io/en/latest/ (accessed 2017-06-16).
[23] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter, *Comparing Community Structure to Characteristics in Online Collegiate Social Networks*, SIAM Review, 53 (2011), pp. 526–543.
[24] A. L. Traud, P. J. Mucha, and M. A. Porter, *Social Structure of Facebook Networks*, Physica A: Statistical Mechanics and its Applications, 391 (2012), pp. 4165–4180.
[25] Tweepy, *Tweepy v3.5.0*, 2017, http://docs.tweepy.org/en/v3.5.0/api.html (accessed 2017-06-16).
[26] Twitter, *REST APIs v1.1 - Twitter Developers*, 2017, https://dev.twitter.com/rest/public (accessed 2017-06-16).