Network Science and Social Media: Analyzing 2013 Glastonbury Music Festival Dataset

Abstract

In this work, we studied the paper ‘Piecing together the puzzle: Improving event content coverage for real-time sub-event detection using adaptive microblog crawling’[1] and have drawn inferences and conclusions from the dataset generated. In the paper, initially the key words related to events were used as a baseline stream and it was proved that comprehensive coverage of sub events is not attained.

They lead to the proposed idea of adaptive crawling technique in which baseline stream is fed to keyword adaptation feature in which similar contents are used to generate additional keywords which are collected as adaptive stream.

The motivation drawn from this paper is relevancy to real time events with more breadth along with depth of event coverage comprehensively in daily life with timely information about real world events. We have used the 2013 Glastonbury music festival twitter dataset for our analysis that is densely packed simultaneous events. The content is derived using sub event formulation and summarization of tweets achieved which is evaluated with Pevent and Revent. The core idea stands relevant to current events which keeps happening in real time.

We have analyzed the 3 different datasets BL (Baseline), AD (Adaptive) and EX (Extra) created from the proposed idea of stream division methodology in which the stream is decomposed into the stream named baseline, adaptive and extra line stream and compared their statistics (using Gephi).

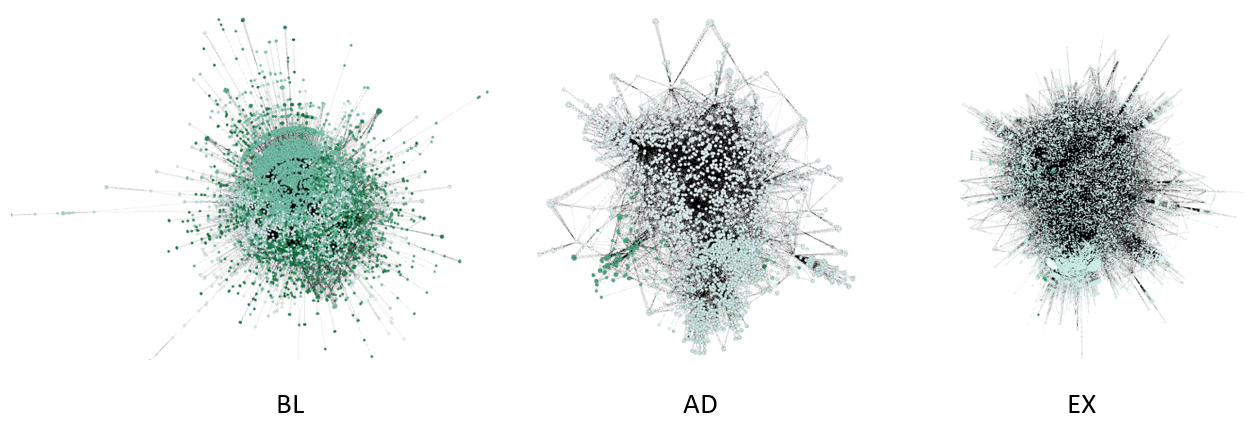
We have then used 3 different community detection algorithms Louvain, Girvan-Newman and Clique and generated the graphs (using Networkx in Python) to identify groups of nodes and their connectivity. Next, we performed sentiment analysis on the whole dataset and the top 20 communities to compare them. Further, we generated random networks for the AD dataset and compared the results to the original data. We then check the robustness of the network by removing nodes and edges and also calculate the pageranks of the network <have to add few more details>.

In most cases, we worked with all 3 different datasets. However, at times we have only dealt with the AD dataset. This is because the data was collected using the content adaptive crawler to collect more information about an event [1] and we could derive more information from this dataset about a sub event.

Introduction

Network Analysis and Graph Theory is already a known concept in the area of social networking. Network Analysis helps us in visualizing multiple data points and drawing insights from a complex set of connections.

This paper concentrates upon insights that can be drawn by analyzing twitter dataset for Glastonbury Music festival and decoding community structure based upon data.

Below we show the different graphs generated from Gephi after initial processing of the datasets. Since the BL dataset consists solely of user specified pre-defined keywords, we see a very scattered graph with no connection among numerous nodes. The AD dataset on the other hand shows a much more connected graph due to adapted keywords. The EX dataset shows partially connections since it is generated by subtracting BL dataset from AD dataset.Fig 1: Undirected networks representing 3 different data streams

The below table shows the network parameters generated from Gephi. For the reasons stated above, BL shows higher network diameter, larger number of communities and greater number of weakly connected components. The graph density has value 0 for BL indicating the network is nowhere near completion. AD shows better results when compared to BL or EX.

|  |  |  |  |
| --- | --- | --- | --- |
| **Network Parameters/Dataset** | **BL** | **AD** | **EX** |
| Average degree | 4.572 | 7.218 | 8.639 |
| Average weighted degree | 13.036 | 110.937 | 46.189 |
| Network diameter | 8 | 6 | 7 |
| Graph Density | 0 | 0.002 | 0.001 |
| Weakly Connected components | 336 | 55 | 19 |
| Average clustering coefficient | 0.856 | 0.889 | 0.873 |
| Average path length | 2.84 | 3.145 | 3 |
| Number of communities | 585 | 64 | 61 |
| Number of Nodes | 9457 | 4555 | 14762 |
| Number of Edges | 21618 | 15643 | 60272 |

Table 1: Network Parameters generated from Gephi on the whole dataset

The rest of this paper is organized as follows: Section 2 introduces not related work from

the perspectives of nodes retrieval of the, event detection and summarization. Then, the overview of

the proposed event monitoring framework is outlined in section 3. Section 4 presents the techniques

and methods employed. Section 5 provides the experimental details and results. Finally,

Section 6 suggests some future <update after final completion of paper>

Related Work

Below is the data overview for AD dataset represented in a node graph.

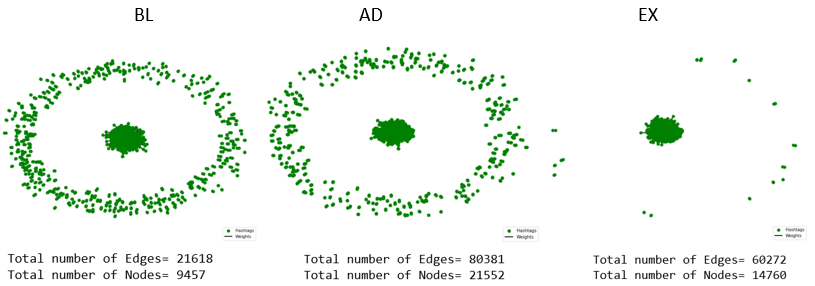


Fig 2: Node graph representations

We then use centrality measure used to identify which nodes are the biggest influencers of the network.

The following image shows the value of the 3 types of centrality

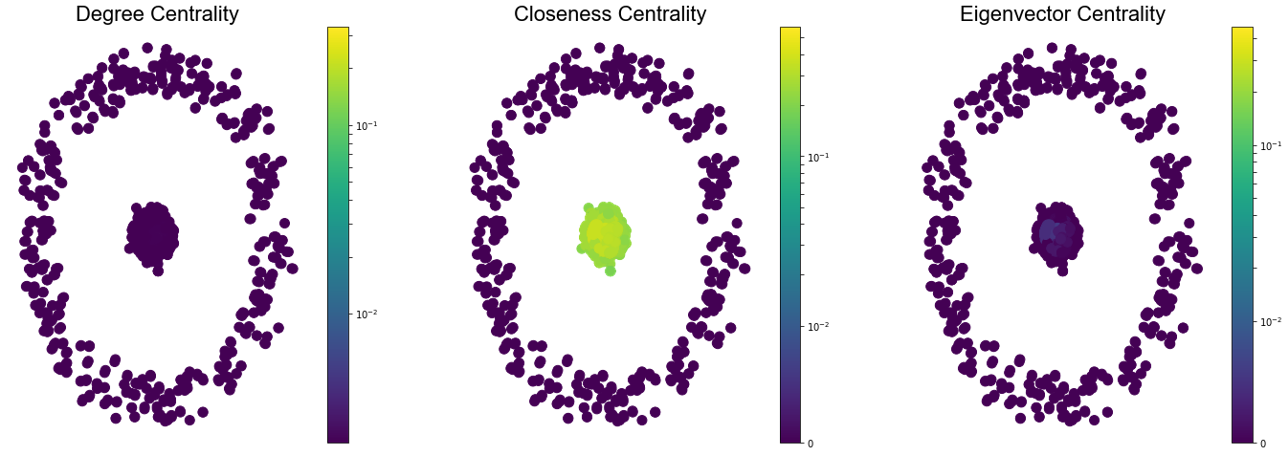


Fig 3: Centrality representation for AD

We find the top 10 most influential nodes with highest centrality values.

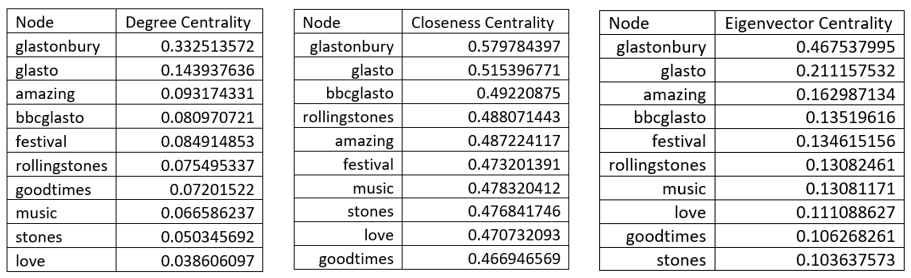


Table 2: Most influential nodes

We can see safely say that glastonbury, glasto, bbcglasto, amazing, rollingstones and festival are the most popular nodes.

Next, we try to form communities based on connectivity and modularity to find nodes that are far more connected than the rest of the network.

1. The Louvain Algorithm [3] works on the principle of partitioning a network into mutually exclusive communities such that the number of edges across different communities is significantly less than expectation, whereas the number of edges within each community is significantly greater than expectation. The Louvain algorithm is one of the most widely used for identifying communities due its speed and high modularity. Modularity values can span from -1 to 1, and the higher the value, the better the community structure that is formed.

We performed the Louvain algorithm on the datasets, and the results for AD are given in Fig 4 below:

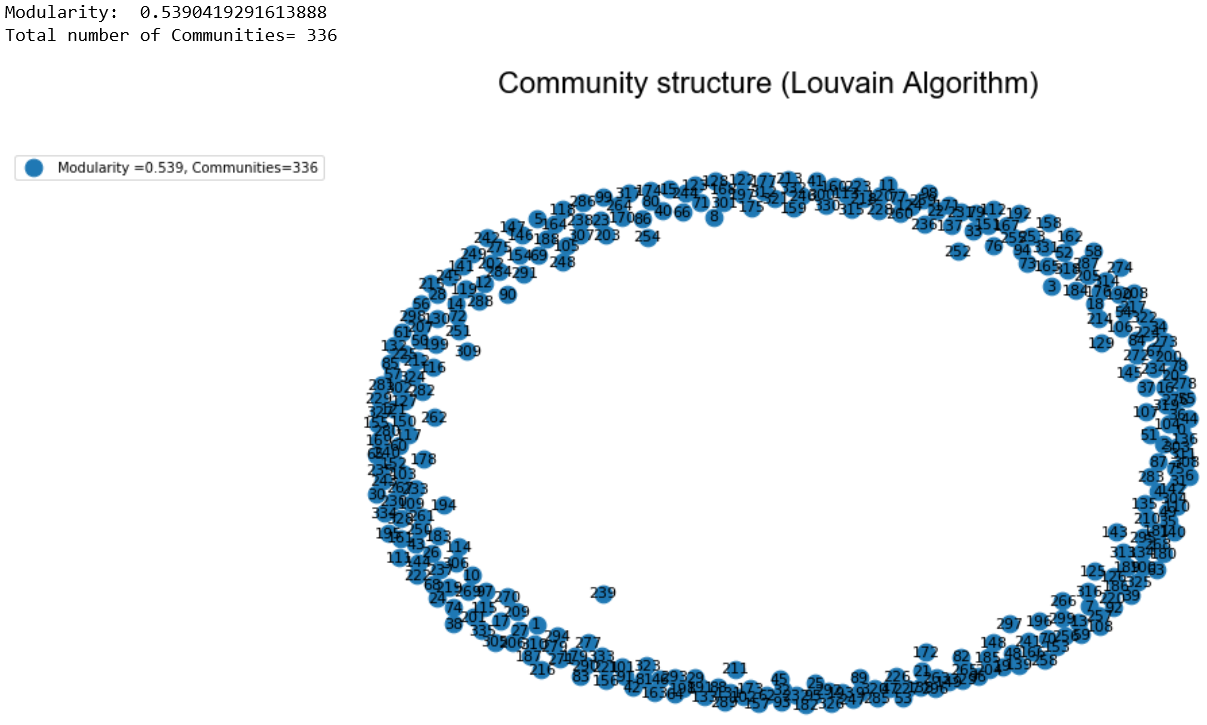


Fig 4: Community using Louvain algorithm

Among the 3 datasets, AD has the highest modularity of 0.539 with 336 communities. We also see that there are only few communities that have very high number of nodes (having the influential nodes) while the others have 5 or fewer nodes.

We then tried to visualize all the non-overlapping communities using different colors and we got the following image illustrated in Fig 5.

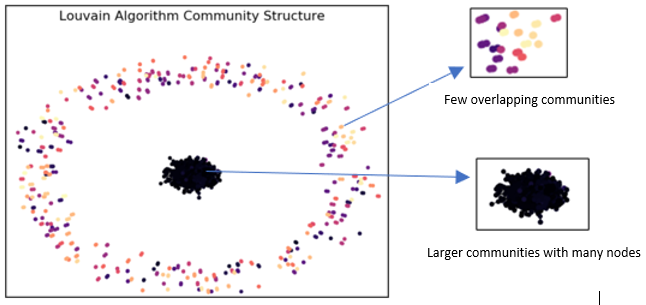


Fig 5: Communities with different colors

2. The Girvan-Newman algorithm gives a very similar solution. However, it is inferior to the Louvain algorithm and gives a much lower modularity of 0.012. The number of communities is found by calculating the betweenness of all edges and then removing the edge with highest betweenness. The process is repeated until no edges remain. We finally get 305 communities. We observe that in this method only 1 community contains very high number of nodes and the rest of the communities majorly contains 3 or less nodes. Given all these factors, we proceeded to perform sentiment analysis on the communities generated using Louvain [3] method. The analysis and results of the sentiment analysis is described in section <45>.

3. We then performed the maximal clique calculation. Cliques are sub-graphs in which every node is connected to every other node. A node can be a member of more than one clique/community hence there is a sense of overlapping structure. As per the Maximal Cliques approach, we find cliques which are not sub-graphs of any other clique. The Bron-Kerbosch algorithm [4] is famous in this aspect, we pick maximal cliques bigger than minimum number of nodes.

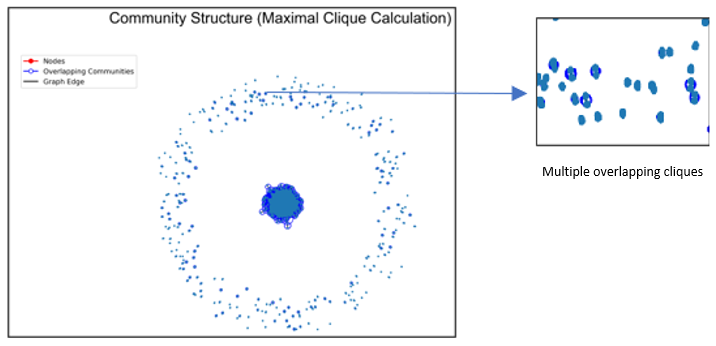


Fig 6: Clique community structure for AD

We then run on this on the datasets. 40918 cliques were formed for the AD dataset. We can see some communities having multiple influential nodes in them. We can also see the interconnectedness between cliques. This can be used to identify a sub-section of communities that are more closely connected than other sets of nodes. This gives us a set of dense and interconnected communities.

Sentiment Analysis is a technique widely used in text mining. Twitter Sentiment Analysis, therefore means, using advanced text mining techniques to analyze the sentiment of the text (here, tweet) in the form of positive, negative and neutral. It is also known as Opinion Mining, is primarily for analyzing conversations, opinions, and sharing of views (all in the form of tweets) for deciding business strategy, political analysis, and also for assessing public actions.

Enginuity, Revealed Context, Steamcrab, MeaningCloud, and SocialMention are some of the well-known tools used for the analysis of Twitter sentiment. R and Python are widely used for sentiment analysis in dataset twitter. We have done, Twitter Sentiment Analysis in Python

This paper focuses on the use of an event stream related to a particular type of event and not the whole Twitter stream. This event stream is collected with a small set of user selected keywords. To explore the detection benefits under different event scenarios, this paperuse for the investigation: the2013GlastonburyMusicFestival (Glastonbury Festival). The Glastonbury festival event consist of densely back multiple sub-events because multiple performances are carried out simultaneously. Both the baseline crawler and the adaptive crawler are triggered by the same set of initial keywords. Since the adaptive crawler identifies extra keywords during the developing of the event, the number of keywords of adaptive crawler is significantly higher than that of baseline crawler, by 117 for Glastonbury Festival. Data was collected during the relevant periods by connecting to Twitter using the public interface provided by Twitter-4J. This interface accesses the freely available 1% stream. All data collected complied with Twitter’s terms of service.

Sentiment Analysis is a technique used in text mining. It may, therefore, be described as a text mining technique for analyzing the underlying sentiment of a text message, i.e., a tweet. Twitter sentiment or opinion expressed through it may be positive, negative or neutral. However, no algorithm can give us 100% accuracy or prediction on sentiment analysis.

As a part of Natural language processing, algorithms like SVM, Naive Bayes is used in predicting the polarity of the sentence. sentiment analysis of Twitter data may also depend upon sentence level and document level.

Methods like, positive and negative words to find on the sentence is however inappropriate, because the flavor of the text block depends a lot on the context. This may be done by looking at the POS (Part of Speech) Tagging.

Sentiment Analysis Dataset Twitter has a number of applications:

**Business**: Companies use Twitter Sentiment Analysis to develop their business strategies, to assess customers’ feelings towards products or brand, how people respond to their campaigns or product launches and also why consumers are not buying certain products.

**Politics**: In politics Sentiment Analysis Dataset Twitter is used to keep track of political views, to detect consistency and inconsistency between statements and actions at the government level. Sentiment Analysis Dataset Twitter is also used for analyzing election results.

**Public Actions**: Twitter Sentiment Analysis also is used for monitoring and analyzing social phenomena, for predicting potentially dangerous situations and determining the general mood of the blogosphere.

**Twitter Sentiment Analysis using Python**:

The dataset being used is of undirected in nature and top twenty communities has been chosen for performing sentiment analysis. As we are aware Twitter is 140 words lines with variety of special characters but conveying the sentiments of the audience.

Following process was followed while performing sentiment analysis.

**Libraries used:**

Pandas 1.0.1

Scikit-image 0.16.2

Scikit-learn 0.22.1

Nltk 3.5

numpy 1.18.1

tensorflow 2.1.0

Keras 2.3.1

Variables were defined in the program which enable the program to read specified column variable. The Dataset has been processed and saved as csv file which was read. Extra columns available in the dataset had been removed. The entire dataset had been split into train and test datasets and saved. Those datasets were dataset imported for execution.

Stop words were identified and removed during initial preprocessing. Tweet is expanded for further processing. User handles has been removed ("@[\w]\*",""). Removing numbers and special characters(^a-zA-Z'#]"," "). Removing URLs((www\.[^\s]+)|(https?://[^\s]+))"), " ")

Removing single characters(r"(^| ).( |$)"). Tokenizing is performed to split the cleaned tweets. Further available stop words are removed. Lemmatisation (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP).Stemming is also a part of queries and Internet search engines.Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras. Sequence classification is a predictive modeling problem where you have some sequence of inputs over space or time and the task is to predict a category for the sequence

Building LSTM model to perform sentiment analysis

Constructing a model.

Creating and adding layers including weight initialization and activation.

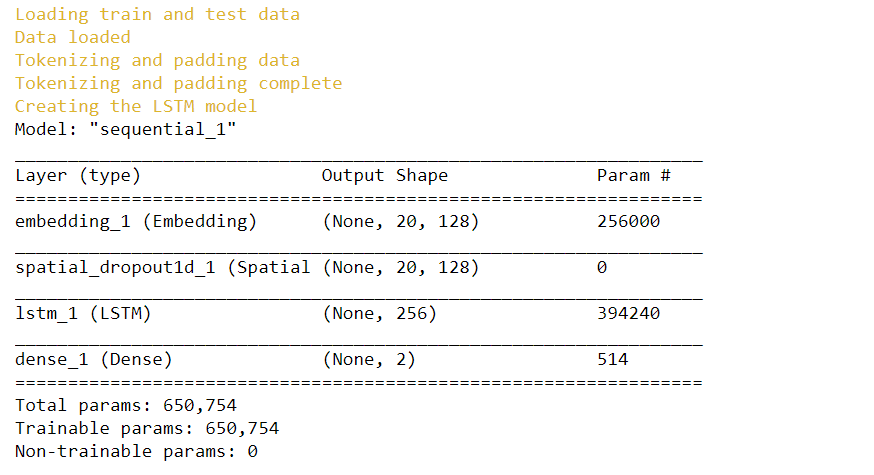
Compiling models including optimization method, loss function and metrics.

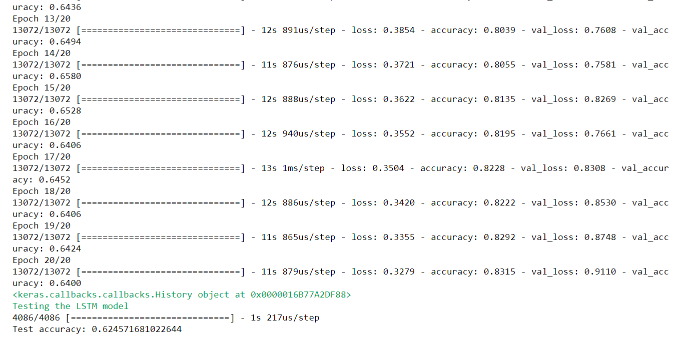
Fitting models including epochs and batch size

Model predictions.

Summarizing the model.

Then the learnt accuracy has been evaluated with more epochs.





**1. How we generate random networks. Why do we compare things to random networks?**

Network science aims to build models that reproduce the properties of real networks. Most networks we encounter do not have the comforting regularity of a crystal lattice or the predictable radial architecture of a spider web. Rather, at first inspection they look as if they were spun randomly. Random network theory embraces this apparent randomness by constructing and characterizing networks that are truly random.

Let us give the overlay of random networks.

G (N, L) Model

* N labelled nodes are connected with L randomly placed links.

G (N, p) Model

Each pair of N labelled nodes is connected with probability p, a model introduced by Gilbert

Hence, the G (N, p) model fixes the probability p that two nodes are connected and the G (N, L) model fixes the total number of links L. While in the G (N, L) model the average degree of a node is simply <k> = 2L/N, other network characteristics are easier to calculate in the G (N, p) model. We will explore the G (N, p) model, not only for the ease that it allows us to calculate key network characteristics, but also because in real networks the number of links rarely stays fixed.

Hence, the G (N, p) model fixes the probability p that two nodes are connected and the G (N, L) model fixes the total number of links L. While in the G (N, L) model the average degree of a node is simply <k> = 2L/N, other network characteristics are easier to calculate in the G (N, p) model. Throughout this book we will explore the G(N, p) model, not only for the ease that it allows us to calculate key network characteristics, but also because in real networks the number of links rarely stays fixed.

A random network consists of N nodes where each node pair is connected with probability p.

To construct a random network, we follow these steps:

1) Start with N isolated nodes.

2) Select a node pair and generate a random number between 0 and 1. If the number exceeds p, connect the selected node pair with a link, otherwise leave them disconnected.

3) Repeat step (2) for each of the N(N-1)/2 node pairs.

The network obtained after this procedure is called a random graph or a random network.

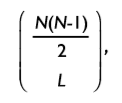
From a modelling perspective a network is a relatively simple object, consisting of only nodes and links. The real challenge, however, is to decide where to place the links between the nodes so that we reproduce the complexity of a real system. In this respect the philosophy behind a random network is simple: We assume that this goal is best achieved by placing the links randomly between the nodes

The probability that a random network has exactly L links is the product of three terms:

1) The probability that L of the attempts to connect the N(N-1)/2 pairs of nodes have resulted in a link, which is p^L

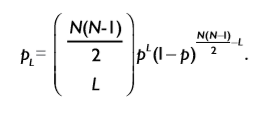
2) The probability that the remaining N(N-1)/2 - L attempts have not resulted in a link, which is 

3) A combinational factor,

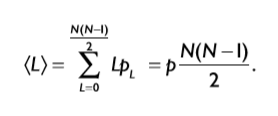


counting the number of different ways, we can place L links among N(N-1)/2 node pairs.

We can therefore write the probability that a particular realization of a random network has exactly L links as



As the above-mentioned equation is a binomial distribution, the expected number of links in a random graph is



Hence <L> is the product of the probability p that two nodes are connected and the number of pairs we attempt to connect, which is Lmax = N (N - 1)/2

Using the expected number of links, we obtain the average degree of a random network

Hence <k> is the product of the probability p that two nodes are connected and (N-1), which is the maximum number of links a node can have in a network of size N.

In summary the number of links in a random network varies between realizations. Its expected value is determined by N and p. If we increase p a random network becomes denser: The average number of links increase linearly from <L> = 0 to Lmax and the average degree of a node increases from <k> = 0 to <k> = N-1.

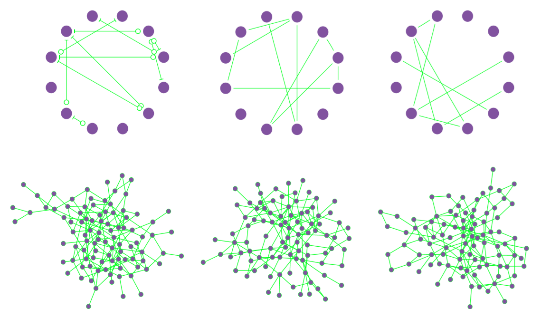
A **random network** framework where the different **networks** are samples from the same probability distribution may be useful for examining such questions. Without any restrictions, a **random network** model is very high-dimensional, but one can make tractable **random network** models through various simplifications.

One advantage of looking at an ensemble of networks defined by a probability distribution is that we can zero in on the influence of different statistical properties. We can imagine taking many samples from the probability distribution and look for properties that are common to most of the samples. In some cases, we might be able to take an ensemble average of a quantity, where we take an average over the probability distribution.

Besides the mathematical convenience, in many cases, looking for properties over a whole network ensemble makes sense in a real world application. For example, despite the fact that people's brains are wired differently, most people are able to perform similar motor tasks, such as picking up an object. If we want to discover the network properties that facilitate such an action, we might not be interested in the network variations among individuals but the features that are common among the individual. A random network framework where the different networks are samples from the same probability distribution may be useful for examining such questions.

**Top Row**

Three realizations of a random network generated with the same parameters p=1/6 and N=12. Despite the identical parameters, the networks not only look different, but they have a different number of links as well (L=10, 10, 8).



**Bottom Row**

Three realizations of a random network with p=0.03 and N=100. Several nodes have degree k=0, shown as isolated nodes at the bottom.

**The dimension of random networks**

For a [random network](https://mathinsight.org/random_network), the random variables are the entries of the [adjacency matrix](https://mathinsight.org/network_introduction#adjacency_matrix). These entries are 0 or 1, depending on whether or not the corresponding edge exists, so the probability distribution is similar to the previous one mentioned above. The dimension of the parameter space describing the probability distribution will be 2^n−1 if we have n different random variables.

If we have a network of N nodes, then the adjacency matrix is an N×N matrix. This matrix has N^2 entries, but if we exclude self-connections and make the network [undirected](https://mathinsight.org/definition/undirected_graph), then there are only N^(N−1)/2 unique entries in the matrix. The probability distribution describing the resulting random network will be in terms of n=N(N−1)/2 different Bernoulli random variables. The dimension of the parameter space is 2^(N(N−1)/2-1.

Therefore, to determine a probability distribution defining an N×N unweighted and undirected random network, we just need to pick 2^N(N−1)/2 different positive numbers (making sure they are all really small so that there sum is no larger than one). For each possible combination of these 2N(N−1)/2−1 different numbers, we have a different probability distribution, hence a different random network model.

Let's say we have a network of N=3 nodes, then we have to pick 2^(3⋅2/2)−1=2^3−1=7 different numbers to specify the network probability distribution. Increasing the network size to N=5 nodes, and the dimension of the parameter space increases to 2^10−1=1023. These numbers are getting large fast. If we stay with tiny networks and just increase the size to N=10 nodes, the dimension of the parameter space jumps to 2^45, which is over 35 trillion. Increase the size to a small network of N=100, and we have to choose over 10^1490 different parameters. We won't even think of what happens if you go beyond a medium size network of N=1000, where the parameter space weighs in at over 10^150,364 dimensions. Given that the number of atoms in the observable universe is [estimated to be around 10^80](http://en.wikipedia.org/wiki/Observable_universe), it's hard to comprehend the size of this number of dimensions.

Reducing the dimension

To develop tractable random network models, it's clear we have to shave off a few dimensions from the parameter space describing the network probability distribution. Otherwise, it will take us awhile to come up with the 10^150,364 number describing the probability distribution for a network of N=1000 nodes.

Statistical homogeneity

In our above calculation, we made the probability distribution completely general. In particular, we allowed all 1000 nodes to be completely different. A dramatic simplification will be to assume that all 1000 nodes are statically identical, or that the network is statistically homogeneous. In other words, we can insist that the probability distribution does not change if we switch the order of the nodes. If we switch node one and node two, we want to make sure that the table of probabilities does not change.

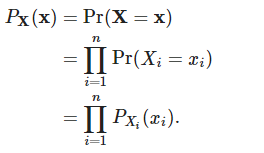
For N nodes, there are N!=N(N−1)(N−2)⋯1 different orderings of the nodes. So it seems we could reduce the size of the dimension by a factor of N! if we insist the probabilities don't change by any reordering. (It actually does not reduce it by quite this much due to complicated symmetries, but let's not worry about this.) Let's say the dimension of the parameter space for NN nodes is reduced to 2^(N(N−1)/2)/N!.

The factorial N! increases very rapidly with N, so dividing by N! should help keep the dimension down. When N=1000, N! is an astronomically large value of around 10^2567. But, unfortunately, 2^(N(N−1)/2) grows much faster. Dividing by N! only reduces the dimension from around 10^150,364 to the not-so-small number of 10^147,797.

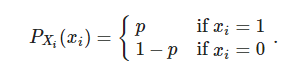
Enforcing statistical homogeneity on the network was not nearly enough to make the dimension tractable. The possible network structures are just too complicated. We need to do something much more drastic to obtain a workable random network model.

Independence

One simplification that will immediately bring down the dimension of the probability distribution parameter space is to assume that the random variables are independent. If the random variables Xi are independent, then we can factor the probability distribution of the vector X= (X1, X2,…,Xn) into many probability distributions for a single variable,



This description is much simpler. If we assume that each random variable   
 Xi is a Bernoulli random variable, then we need just one parameter (the probability pi that it is 1) to determine the probability distribution PXi(xi):



For the probability distribution of the n variables in X, we just need n different parameters, way less than the 2n−1 we needed without the independence assumption.

This assumption makes it tractable to write down a probability distribution even for large networks. For an undirected network of N nodes, we will have n=N(N−1)/2n independent Bernoulli random variables corresponding to each possible location for an edge. We'll need to pick just N(N−1)/2 parameters (the probabilities pij that there is an edge between each pair of nodes i and j). If N=1000, this is 499,500 different numbers, a few less than 10^150,364 we had earlier.

Still 499,500 is still a bit large. We can reduce this further by assuming statistical homogeneity, which in this case is assuming the probability pij of an edge does not depend on the node labels i and j. Now we just need to pick a single parameter pp, the probability that any edge exists. We have reduced the parameter space down to one dimension!

What we've come up with is the Erdös-Rényi random network model, the [simplest type of random network model](https://mathinsight.org/simplest_type_of_network). This model is so natural, that people often refer to these networks as simply “random networks,” and may view deviations from the Erdös-Rényi model as “non-randomness. “

This model stays one-dimensional even if we extend it to [directed](https://mathinsight.org/definition/directed_graph) graphs and is easy to generate. We can also extend the model to populations of different types of nodes, allowing the probability of a connection to depend on the populations of the nodes involved. As long as we keep the independence assumption, the model is easy to understand, and it is a simple matter to generate the networks.

Random graphs are used not because all the properties of networks are described by its parameters, like distribution of links, length of the longest connected set, but because the number of possible networks is enormous and because we need to cluster them in types. For each type (here random network with certain properties) we can simulate some social interaction. A big question still remains what part of interaction can be described by random network parameters and what part by its particular realization. Statistics will just tell us that the rest is random noise. But the difference from other statistical applications is that origin of network is not essentially random. For every social application (propagation of diseases, propagation of gossips, etc) the separation of social outcome into explained and random component may differ. But still this is the best we can do. In fact, we often cannot measure all topological structure of real network and have to derive result based on approximate values and topology.

Similarly, if you have a real life network you can learn a lot by comparing it to a random network - but a relevant random network, namely one that has the same mean connectivity for example (i.e. each node has the same number of neighbours on average). For example, in a social network the number of triangles is consistently higher than in a random network with the same number of links per site. This corresponds to the fact that if you have two friends it is likely that they will be friends too. This insight is not huge but it captures something important regarding social networks, that your exposure to new people is mediated by your existing social links and not created at random - you get to know new people usually because someone you know already knows them. This is just an example - but I hope you see where it is going.

The main issue is matching your real-world network with a relevant random graph, technically speaking, with a proper null-model. A quite general framework is provided by the so-called "the configuration model", namely an ensemble of random graphs that realize a certain degree distribution. In other words - I think that if you compare your real-life network with a random network with the same degree sequence for a certain property you can learn how probably certain features are.

**Robustness of twitter data of 2013 Glastonbury music festival**

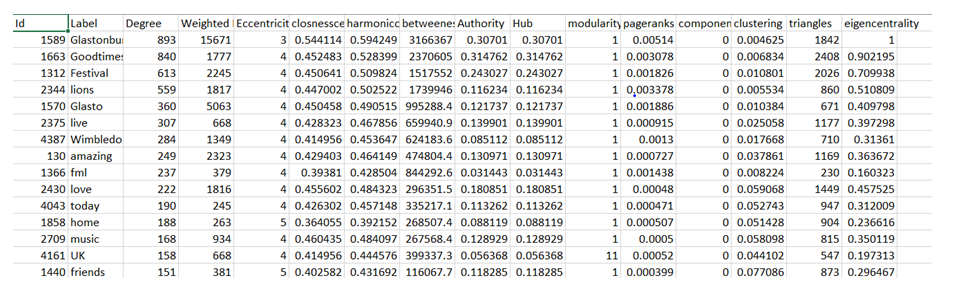
Given a social network, which of its nodes have a stronger impact in determining its structure is always a concept of deserving attention. More formally: which node-removal order has the greatest impact on the network structure is being experimented well before this implementation. We approach this well-known problem following the well research oriented papers in a setting that utilize social networks, using datasets of 2013 Glastonbury music festival. Our experiment highlight the robustness of networks in the structure of social networks and random network.

One of the most important notions that researchers have been trying to capture is “node centrality”: ideally, every node (often representing an individual) has some degree of influence or importance within the social domain under consideration, and one expects such importance to be reflected in the structure of the social network; centrality is a quantitative measure that aims at revealing the importance of a node. Among the types of centrality that have been in practice, many have to do with shortest paths between nodes; for example, the betweenness centrality of a node N is the sum, over all pairs of nodes u and v, of the fraction of shortest paths from u to v passing through N. The role played by shortest paths is justified by one of the most well known features of complex networks, the so-called small world phenomenon. A small-world network is a graph where the average distance between nodes is logarithmic in the size of the network, whereas the clustering coefficient is large than in a random Erdos Renyi graph with the same size and average distance. Here, and in the following, by “distance” we mean the length of the shortest path between two nodes. The fact that social networks (either electronically mediated or not) exhibit the small-world property is known at least since Milgram’s famous experiment and is arguably the most popular of all features of complex networks.

We thus consider a certain ordering of the nodes of a graph (that is supposed to represent their “importance” or “centrality”). We remove nodes (and of course their incident arcs) following this order, until a certain percentage of the nodes have been deleted . Finally, we compare the network parameters of Glastonbury data and average of three random networks with same number of nodes. The chosen ordering is considered to be a reliable measure of centrality ,which is sufficient to delete a small fraction of important nodes to change the structure of the network. In this work, we applied the described approach to a number of both networks, considering ordered removal of nodes, and obtained the following results: In all complex networks which is used here, the removal of a limited fraction of nodes does not change the key network parameters which will be explained below.

Average degree is simply the average number of edges per node in the graph. It is relatively straightforward to calculate. Average Weighted Degree is average sum of weights of the edges of nodes. The graph is designed in such a way that, weight of an edges represents, how many times that edges is traversed between a pair of nodes. If weight of node is higher, it means it has been visited many times than any other low weight degree node.

Network diameter is the shortest distance between the two most distant nodes in the network. In other words, once the shortest path length from every node to all other nodes is calculated, the diameter is the longest of all the calculated path lengths. Graph density is measured as the number of possible or potential connections (i.e., edges), over the number of actual connections . Density values range between zero and one, and can be thought of as the percent of all possible edges that are realized.

Modularity is one such concept that provides information about how the communities are formed within social network. Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random. A connected component is a subset of nodes such that there are two conditions this set of nodes satisfy. First, every node in the subset has to have a path to every other node in the subset. Average clustering coefficient is the clustering coefficient (Watts-Strogatz), when applied to a single node, is a measure of how complete the neighborhood of a node is. When applied to an entire network, it is the average clustering coefficient over all of the nodes in the network. Average path length is a concept in network topology that is defined as the average number of steps along the shortest paths for all possible pairs of network nodes. It is a measure of the efficiency of information or mass transport on a network.

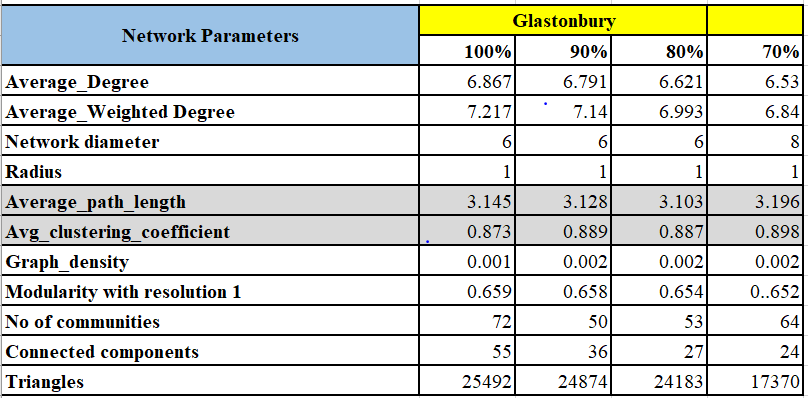
The idea of grasping information about the structure of a network by repeatedly removing nodes out of it is not new: Albert, Jeong and Barabasi study experimentally the variation of the diameter on two different models of undirected random graphs when nodes are removed either randomly or in “connectedness order” and report different behaviours. More recently, node-centrality measures that look at how some graph invariant changes when some vertices or edges are deleted have been studied by identifying nodes that maximally disconnect the network.

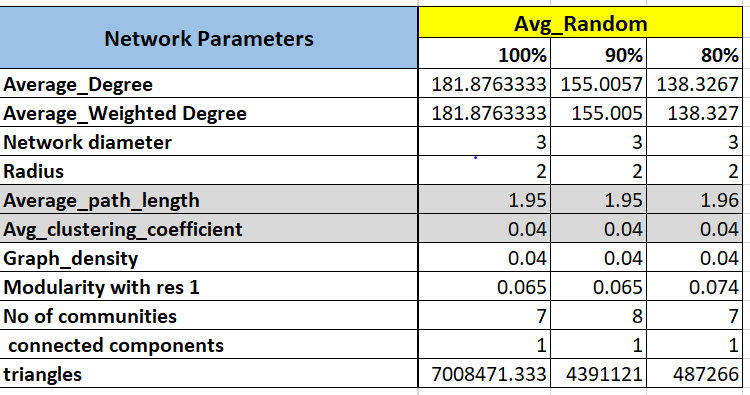
We considered several different strategies for removing nodes from a graph. Some of them embody actually significant knowledge about the structure of the graph, whereas others are very simple (or even independent of the graph) and will be used as baseline. As we handle a undirected network ,we requires a symmetric graph . Initially we pick random nodes and remove them from the graph. It is important to test against this unusual practice as we can show that the behaviour we observe are due to the peculiar choice of nodes involved, and not to some generic property of the graph. We later resorted to large degrees first and we remove nodes in decreasing degree order. This strategy is an obvious baseline, as degree centrality is the first shot at centrality in a network. Social networks are much more resistant to node removal. There is no strict clustering, nor definite hubs, that can be used to eliminate or elongate shortest paths. This is not surprising, as networks emerging from social interaction are much less drafted as we can say there is less page mappings or definite hierarchy.

**Experiment results:**

Number of nodes: 4555

Number of edges: 15643





We have removed the percentage of nodes in the order of 10,20 and 30 and observed the robust behaviour of Glastonbury dataset. The average degree and path length display least change and remain well connected.We have explored experimentally the results of node removal. We have confirmed some of the experimental results that appeared in the previous papers. In particular, we have shown the lucid structural difference between social networks and random networks, and that it is important to test node-removal strategies until a significant fraction of the edges have been removed. Probably the most important conclusion is that “scale-free” models, which are currently proposed for social networks need more analysis. For more intuitive analysis, the above conducted practice can only make sense as long as they are adopted as baselines. For instance, checking whether node removal alters the clustering coefficient would not be so interesting, because the clustering coefficient of each node depends only on the structure of the neighbourhood of each node. Thus, by removing first the nodes with high coefficient it would be trivial to make the clustering coefficient of the graph decrease quickly.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Terminology

**Degree**: Measures number of incoming connections  
**Closeness**: Measures the minimum number of steps can one node connect to others in the network  
**Eigenvector**: Measures a node’s connection to those who are highly connected. A node with a high score will be someone who is influencing multiple nodes (who in turn are highly connected) and is exercising control behind the scenes.

**Modularity**:

**Nodes**: (hashtags):

Conclusions

Community detection is a very active field in complex networks analysis, consisting in identifying groups of nodes more densely interconnected relatively to the rest of the network. The existing algorithms are usually tested and compared on real-world and artificial networks

References

1. Tokarchuk L, Wang X, Poslad S (2017) Piecing together the puzzle: Improving event content coverage for real-time sub-event detection using adaptive microblog crawling. PLoS ONE 12(11): e0187401. <https://doi.org/10.1371/journal>. pone.0187401
2. Newman, M.E.J., Girvan, M.: Finding and Evaluating Community Structure in Networks. Phys Rev E 69, 026113 (2004)
3. Fast unfolding of communities in large networks  
   Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre  
   Journal of Statistical Mechanics: Theory and Experiment (10), P10008, 2008.
4. E. A. Akkoyunlu, The enumeration of maximal cliques of large graphs,SIAM J. Computing 2:1–6 (March 1973).
5. C. Bron and J. Kerbosch, “Algorithm 457: Finding all cliques of an undirected graph,”Commun. ACM 16:575–577 (September 1973).
6. Tdedwtrk
7. A random graph generation algorithm for the analysis of social networks James F. Morris, Jerome W. O’Neal, and Richard F. Deckro
8. Revisiting the Small-World Phenomenon: Efficiency Variation and Classification of Small-World Networks
9. Tore Opsahl1, Antoine Vernet2, Tufool Alnuaimi2, and Gerard George
10. Network Science Random Networks Barabasi
11. ttrfkeurk