Network Analysis of Google trending keywords

Sara Bertoldo
sarabert96@gmail.com
Student ID: 588361

Pouria Faraji
p.faraji@studenti.unipi.it
Student ID: 585089

Chiara Spampinato
spampinatochiara19@gmail.com
Student ID: 589510

ABSTRACT

In this paper, we will use graph theory to study the structure of relations between semantically proximate words obtained by Google Trends top keywords of 2019. In contrast to ordinary networks, in our graph each node represents a word, shaping a semantic network.

The phases of the work are crawling of data, basic network analysis, community discovery, supervised link prediction, graphlets count estimation and an improved version of link prediction based on word meaning.[1]

KEYWORDS

Social Network Analysis, Google Trends, Semantic Network

ACM Reference Format:
Sara Bertoldo, Pouria Faraji, and Chiara Spampinato. 2020. Network Analysis of Google trending keywords. In *Social Network Analysis '20*. ACM, New York, NY, USA, 10 pages.

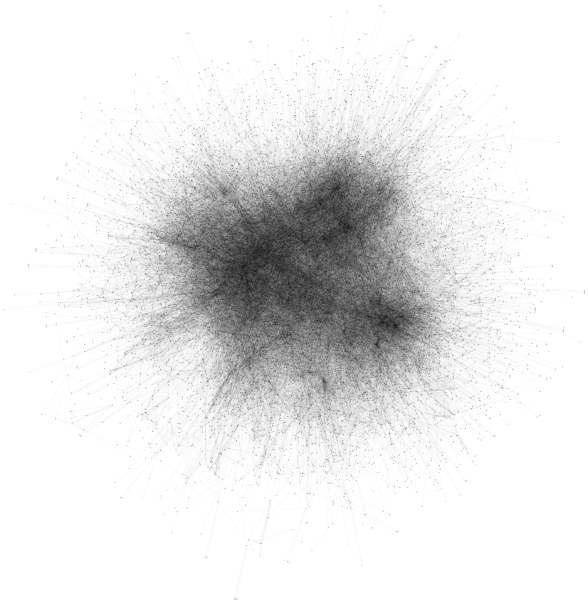

Figure 1: Network visualization

[1]**Project Repositories**
Data Collection:
https://github.com/sna-unipi/data-collection-2020_bertoldo_faraji_spampinato
Analytical Tasks:
https://github.com/sna-unipi/network-analysis-analytical-tasks-2020_bertoldo_faraji_spampinato
Report:
https://github.com/sna-unipi/project-report-2020_bertoldo_faraji_spampinato

SNA '20, 2019/20, University of Pisa, Italy

1 INTRODUCTION

In this report we will analyze the network built with Google trending keywords. We crawled related search terms starting from the top keywords from the year 2019. The network is a directed and unweighted graph, containing 18002 nodes and 70639 edges. The link between words is created if they relate semantically based on Google Trends suggestions. We mainly carried out the project using Python in Google Colab and Jupyter environment, with the help of the libraries NetworkX, CDlib, NDlib and Gephi to visualize the graph.

Subsequently we will present each phase of our study. We started with crawling of data, followed by basic network analyses consisting of degree distribution, connected components, path, clustering coefficient, density and centrality analyses performed also on artificial graphs. The succeeding work studies community discovery, supervised link prediction, graphlets count estimation and an improved version of link prediction based on word meaning.

2 DATA COLLECTION

Our data consist of a series of words connected together depending if they correlate with each other (correlate that is to say that users who searched for word A also searched for word B, hence there exists a directed link between A and B), using Google Trends data. We chose 2019 as study year for temporal proximity reason. To collect data, we started with the 10 top searched words and gradually moved to their 5 top related words, until reaching enough nodes.

Crawling Methodology and Assumptions

Pytrends library[2] was used to crawl Google Trends data. Initially we obtained the first top 10 trending terms in 2019:

(1) India vs South Africa
(2) Cameron Boyce
(3) Copa America
(4) Bangladesh vs India
(5) iPhone 11
(6) Game of Thrones
(7) Avengers: Endgame
(8) Joker
(9) Notre Dame
(10) ICC Cricket World Cup

The process then consisted in obtaining the top 5 related terms for each keyword, removing the repetitions and until reaching at least 15k nodes. Finally, the nodes and the edges have been saved in csv files.

3 NETWORK ANALYSIS

From the obtained data we created a network, visualized in Figure 1 using Gephi, Yufan Hu Layout.
The network characteristics are listed below:

Table 1: Characteristics of RW network

Number of Nodes	18002
Number of Edges	70639
Weighted	No
Directed	Yes
Average Degree	3.924
Density	$2.18 \cdot 10^{-4}$
Number of self-loops	32
Average Clustering Coefficient	0.167

In each of the following analyses, we compared the Real World network (RW) with Erdos-Renyi (ER), Barabasi-Albert (BA), Watts-Strogatz (WS) and Configuration Model (CM) networks. To create the models we used the predefined algorithms. With CM the parameters were the *in* and *out* degree, with ER the parameters were the number of nodes and edges, as well as the directed type, while for both WS and BA we set the number of nodes and some numeric values identified with a process of trial and error looking for the most similar number of edges as result.

Every time we ran the code to create the artificial models, they were built from scratch, so our results changed. To avoid this issue, once we generated our models, we saved them in graphml files, uploading them in the following executions.

Table 2: Nodes and Edges of the networks

	RW	ER	WS	BA	CM
Num of Nodes	18002	18002	18002	18002	18002
Num of Edges	70639	70639	72008	71992	70573

Degree Distribution

Comparing the degree distribution of our networks we noticed that the Configuration Model is the most similar one. This happens because CM is a random network model that completely relies on keeping the same degree as the RW network. Beyond that, the RW network degree trend is also very similar to a BA model, which therefore follows the subsequent degree distribution formula and can be assumed as a scale-free network:

$$P(k) \sim Ck^{-\gamma} \tag{1}$$

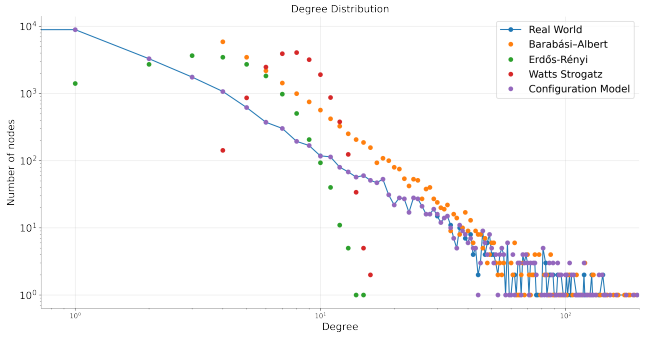

Figure 2: Degree Distribution

In RW network the higher the degree, the lower the number of nodes. The node's highest degree is 583 (labeled with the word "Casting"), and the minimum is 1; therefore there are no nodes with degree 0, resulting in no isolated nodes. The average degree is 3.924 obtained dividing the number of edges by the number of nodes (classic directed graph formula).

[2]Pytrends library description: https://pypi.org/project/pytrends/

Connected Components

For the purpose of comparing the connected components, we treated all the networks as undirected graphs. The five networks are composed by a different number of connected components:

- RW: 2
- ER: 6
- BA, CM, WS: 1

Specifically, our Real world network has 3865 strongly connected components and 2 weakly connected components.

Path Analysis

To calculate the Average shortest path for RW, CM, and ER we considered the weakly connected components, differently from WS and BA where we used the basic connected components. The biggest connected component of each graph has an average shortest path as represented in Table 3, while all the other connected components have average shortest path 0 (missing in the table).

Table 3: Avarage Shortest Path

	RW	ER	BA	WS	CM
# nodes big comp.	17996	17996	18002	18002	18002
Avg Short Path	7	6	4	5	6
# Conn. Comp.	2	6	1	1	1

Clustering Coefficient

For what concerns the clustering coefficient, which represents the clustering degree of a network, we obtained the following results:

Table 4: Clustering Coefficients

	RW	ER	BA	WS	CM
min	0.000	0.000	0.000	0.000	0.000
max	1.000	0.083	0.500	0.833	0.167
mean	0.167	0.0002	0.005	0.082	0.001
stdev	0.173	0.002	0.024	0.078	0.006

The RW network has similar values to the WS model. Usually, random models have vanishing clustering coefficient for large size; Watts-Strogatz model instead, captures large clustering coefficient and short distances, creating a small world model. In a small world system, it is easy that all or most nodes are linked together. Indeed, in our RW network, we do not have isolated nodes, demonstrating that the results obtained are meaningful.

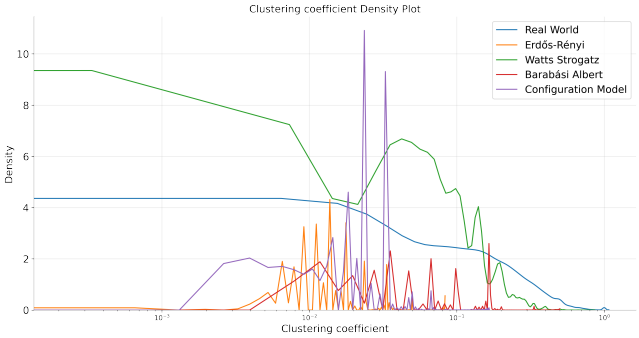

Figure 3: Clustering Coefficients distribution

In the RW network, the clustering coefficient is well distributed: an equal number of nodes have a clustering coefficient in the range 0 to 0.01, the others have a clustering coefficient in the range 0.01 to 1, in which lowering the number of nodes will increase the coefficient. All the other networks have a fluctuating trend with a clustering coefficient never reaching 1.

Density Analysis

The obtained values are:

- RW: $2.180 \cdot 10^{-4}$
- ER: $2.180 \cdot 10^{-4}$
- BA: $4.443 \cdot 10^{-4}$
- WS: $4.444 \cdot 10^{-4}$
- CM: $2.178 \cdot 10^{-4}$

The density of RW network is lower than the BA and WS ones, but it is akin to the remaining. We can conclude that our network has a density value similar to random networks. The difference in the values is given by their diverse topology.

Degree Centrality

The node with the highest degree is "Casting" (583), which is the most important node, followed by "Lyrics" (327) and "Video game" (279). From figure 4, we can see that the WS and ER models have much higher values than others, in fact we can barely see these curves.

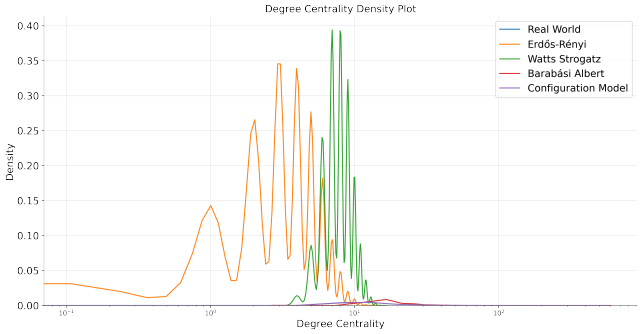

Figure 4: Degree Centrality distribution

Inspecting the remaining ones in figure 5, we notice that RW and CM have a very similar behavior, as already explored in the Degree distribution subsection 3.

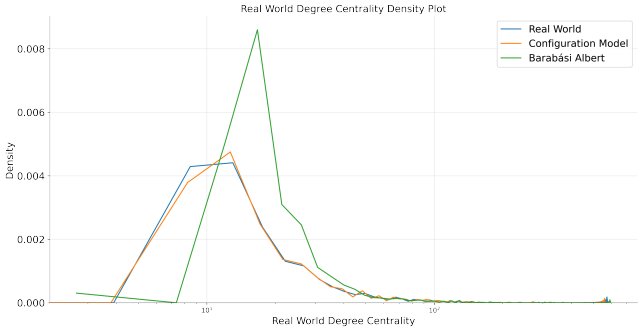

Figure 5: Degree Centrality distribution (particular)

Connectivity - Based Centralities

In all the following Centralities the values of the networks follow the same pattern: the RW and the CM curves are tantamount, the BA model differs from them but it still has comparable values, while WS and ER are totally different, having a behaviour that resembles each other.

Eigenvector Centrality. The nodes with the highest Eigenvector value are "Cricket" (0.275), "India national cricket team" (0.269), "ICC Cricket World Cup" (0.197). As stated by definition, these nodes have a high value because their linked ones have it too. We can presume that all these three nodes are connected together, furthermore, the most searched string of 2019 was "India vs South Africa" which was a Cricket match.

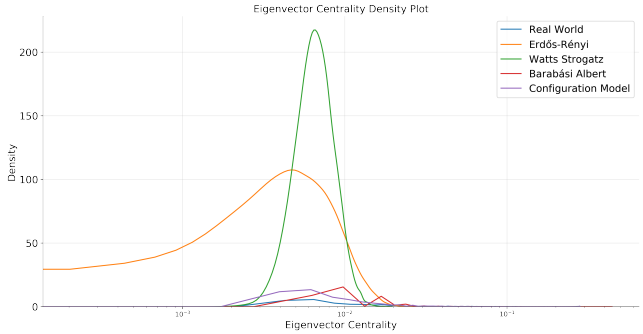

Figure 6: Eigenvector Centrality distribution

Page Rank Centrality. The PageRank of a node depends on the number of links it receives, the link propensity of the linkers, and the centrality of the linkers. Our top three Page Rank nodes are "Casting" (0.00461), "United States" (0.00433), "Car" (0.00307).

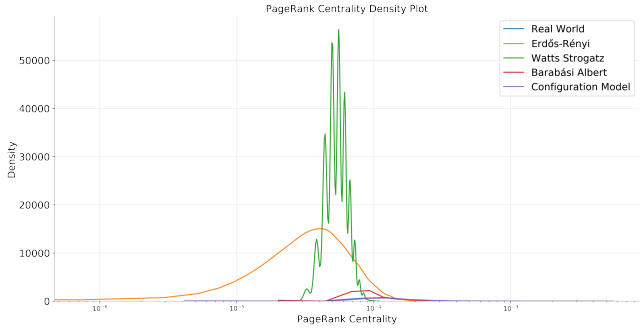

Figure 7: Page Rank Centrality distribution

Katz Centrality. Katz Centrality measures the degree of influence of a node taking into account the total number of walks between a pair of nodes. In our case, the highest nodes are "Casting" (0.255), "United States" (0.149), "Video Games" (0.134).

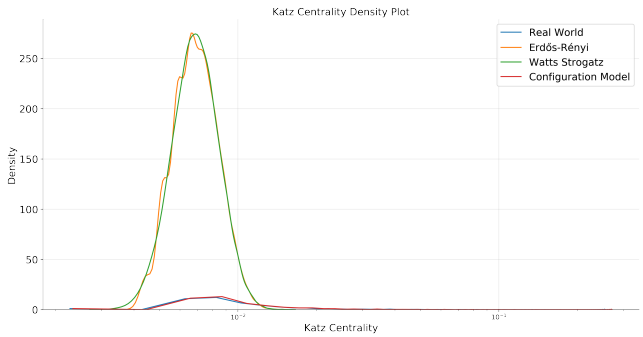

Figure 8: Katz Centrality distribution

Geometric Centrality

Closeness Centrality. The node with the highest closeness value is the most central one, i.e. has the lower average distance with all the other nodes. In our case, the most central are "United States" (0.197), "Company" (0.181) and "Casting" (0.180). If the value is 1, it means that this node is directly connected to all other nodes, which is impossible in our network given the fact that the highest degree value is 583 instead of 18002-1.

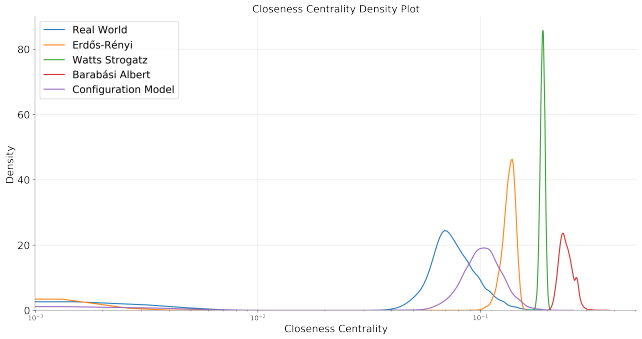

Figure 9: Closeness Centrality distribution

RW network has lower values compared to the others, and all of them has approximately a normal distribution.

Harmonic Centrality. The node with the highest Harmonic Centrality has the lower harmonic mean of the shorted paths distances from a given node to all others and therefore is the node with a trade-off between its centrality and its degree. We expect them to not deviate from the upper one, and in fact the greater nodes are "United States" (3963.7), "Casting" (3917.2), "Company" (3586.9).

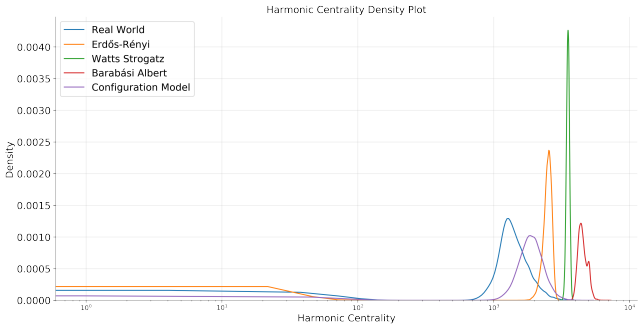

Figure 10: Harmonic Centrality distribution

Here as well, the Harmonic Centrality has lower values in our network. The curves resemble normal distributions and WS network has a much narrow range.

Betweenness Centrality. The node with the highest Betweenness Centrality is the one gone through the higher number of shortest paths. In our case "Casting" ($1.32 \cdot 10^7$), "Death" ($9.99 \cdot 10^6$), "Cause of death" ($7.52 \cdot 10^6$).

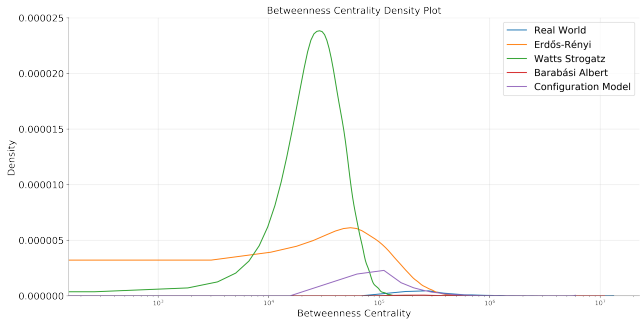

Figure 11: Betweenness Centrality distribution

The values of RW goes roughly from 10^5 to 10^7, with a peak before 10^6. Its distribution is largely different from the one of the others.

Final Considerations. Comparing the distributions of Degree and Connectivity-Based Centralities, we can notice that they are alike.

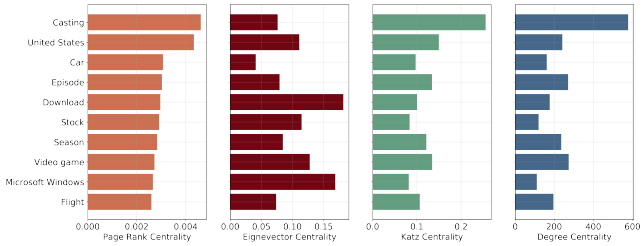

Figure 12: Top Ten Connectivity-Based and Degree Centrality

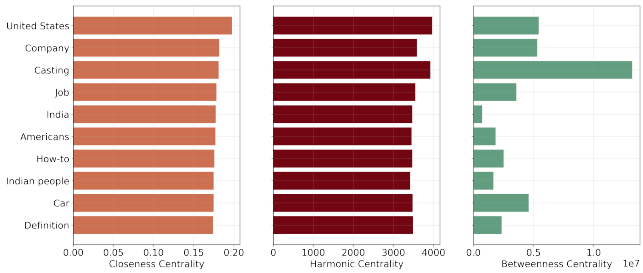

Figure 13: Top Ten Geometric Centrality

Analyzing our network, the most important words are nearly always the same ones. The most salient node is "Casting", which appears in the top three words of each measure,

except for Eigenvector centrality, whose nodes are very peculiar. Excluding these nodes, all the other meaningful terms are very general ones. Such behaviour is justifiable since general words can be easily connected with other terms, unlike specific ones.
Eigenvector Centrality was the only method that identified the semantic field of the most searched term of 2019. One possible explanation is that the first node from which we started the data crawling (the most searched word) is the most central one and that all other nodes developed around it. The Geometric Centrality might not give us remarkable results since how a user surf on the net follows a semantic logic instead of the shortest distance one. For example, starting from United States, it is easier to look for Trump or America instead of Casting which is a very linked word but with no semantic connection at all.

4 COMMUNITY DISCOVERY

Community Discovery aims to identify communities that are hidden behind a complex network structure. We used the algorithms Demon/Angel, K-clique, Louvain, and Label propagation using CDlib. Our network is a directed graph, but to carry out this task, we treated it as undirected.

Evalutating Partition Quality

Table 5: Evaluation of algorithms (mean values)

	K-Clique	Label prop	Louvain	D/A
# comm.ties	1377	1727	36	836
# nodes inv.	13836	18002	18002	13329
degree	3.52	2.78	5.33	5.49
density	0.25	0.17	0.02	0.06
size	12.50	10.42	500.05	50.46
# edges	43.07	23.20	1447.77	148.94
cut ratio	$1.7 \cdot 10^{-4}$	$9.2 \cdot 10^{-5}$	$5.4 \cdot 10^{-5}$	$1.1 \cdot 10^{-4}$
modularity	0.046	0.041	0.053	0.043
conduct.	0.47	0.38	0.14	0.28

As is evident in Table 5, high value in the number of communities of the Label Propagation means that each node has few neighbours. If it had more, the node label would be propagated to more nodes, leading to a fewer number of communities. As regards Louvain, it is in its nature to find a lower number of communities because it tends to avoid small and dense communities.

K-Clique and Demon/Angel are overlapping algorithms, therefore the total number of nodes involved in the communities is less than Label propagation and Louvain. It means that some nodes may be part of multiple communities and some may be left apart.

The better result for the degree is given by the Demon/Angel algorithm.

K-Clique starts with fully connected networks and then identifies the communities, therefore we expect higher density for its communities with respect to other algorithms. Louvain, as mentioned before, avoids dense communities, so it has the lowest density value.

The size is self-explanatory regarding the number of communities: the higher the number of communities, the smaller the number of nodes.

In the same manner, the bigger the community, the higher the number of edges. Louvain has the biggest communities and thus has a higher number of edges.

K-Clique has the highest number of cut ratio, it means that its communities are more intra-connected, while in Louvain, which has a lower cut ratio, communities are more isolated.

Modularity value is positive if the number of edges within groups exceeds the number expected on the basis of chance. All the algorithms gave us positive values, although pretty low.

The best result for the conductance is obtained with Louvain algorithm.

Comparing partitions

Since Normalized Mutual Information (NMI) works under the conditions of complete coverage of nodes, it has been applied only on Label propagation and Louvain. The obtained result is 0.56. Since the higher the NMI the more similar the compared partitions are, in a range between 0 and 1, we can say that the partitions are not as similar to be considered aligned, but neither as different to be treated as fully separated.

Table 6: NF1 score matrix

	Label Prop	D/A	K-Clique
Louvain	$6.1 \cdot 10^{-4}$	0.01	$5.7 \cdot 10^{-5}$
Label Prop		0.06	0.04
D/A			0.02

The results from NF1 score are shown in Table 6. From these values, we can see that actually Label Propagation and Louvain are more different than what we expected reading NMI result. Also, we can't say any two pairs of algorithms are similar due to their closeness of NF1 value to 0.

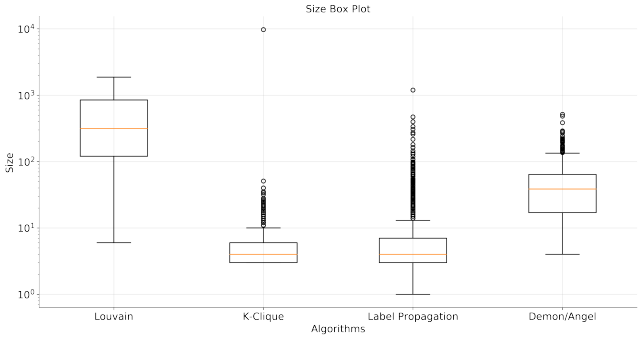

Figure 14: Boxplot of sizes

The boxplot of Figure 14 shows the distribution of sizes of the algorithms. The communities in Louvain have a wider range of number of nodes than the others. The majority of communities for K-Clique, Label Propagation, and Demon/Angel have a lower number of nodes, distributed in a short-range, treating communities with a higher number of nodes as outliers.

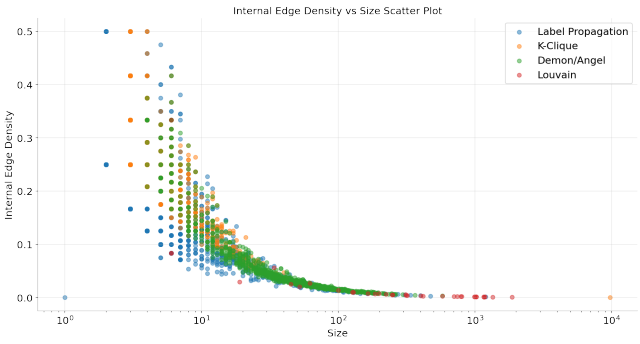

Figure 15: Scatter plot Size vs Internal Edge Density

As can be seen from Figure 15, the higher size of communities correlates with a lower value of edge density.

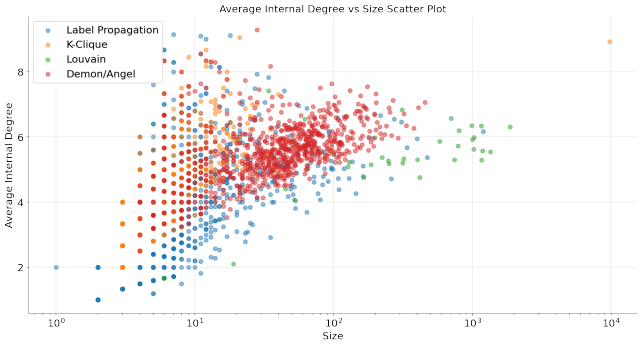

Figure 16: Scatter plot Size vs Average Internal Degree

As there are a low number of nodes in communities, average degree has fixed values, on the other hand, as the number of nodes increase, average degree varies more, resulting in an irregular distribution of dots.

5 LINK PREDICTION SUPERVISED

The task consists in predicting links between nodes in the graph. For time complexity reasons, we had to take into account only a part of the graph (5000 nodes), considering only its biggest connected component. To check whether two nodes are linked, an adjacency matrix has been built, used to create a list of links that can be removed, avoiding to split the graph into multiple connected components.
We then created a graph in which we removed the omissible links, applying node2vec[3] algorithm to extract node features from this graph. We trained the model on the graph obtaining the features of every node. To validate the performance of our model we split the data into training (70%) and test (30%). Finally, a Logistic Regression model [4] predicted the links. We repeated this process for the remaining nodes. The results in all the executions were approximately the same, obtaining an average ROC AUC score of 0.50.

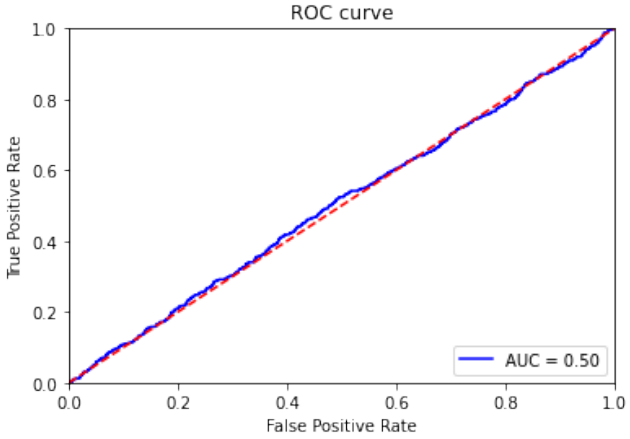

Figure 17: Roc Auc curve

As we can see in Figure 17 our result is not very good, so the prediction will not be reliable. If we had used an unsupervised method the result would have been even worse, since these algorithms have limited performances.

[3]Node2Vec project description: https://pypi.org/project/node2vec/
[4]Logistic Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

6 GRAPHLETS

Graphlets are small, connected, non-isomorphic induced subgraphs of a large network. The aim of the task is to design an algorithm to estimate the number of graphlets of size 3 and 4 in our network. The different shapes of the graphlets are shown in Figure 18

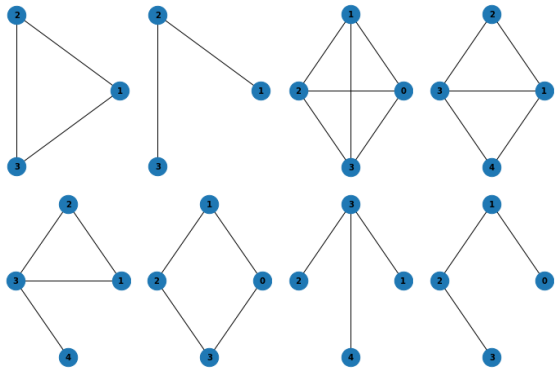

Figure 18: In order: Triangle, Two Star, Four Clique, Chordal Cycle, Tailed Triangle, Four Cycle, Three Star and Four Path

We define a *sampling factor* which is the fraction of the number of edges over the total number of edges. At first, we sample random edges in the graph based on the sampling factor and then for each edge we take all neighbouring nodes connected to the current edge and then we tried to count the number of target graphlets in all combinations of three and four nodes in the neighbouring list, according to the types of graphlets.

Finally, after all edges have been iterated the final count value is divided by the sampling factor so we have estimated count for the whole graph.

The sampling factor represents the percentage of the graph we are searching for the graphlets. 0.5 means looking at only 50% of the total number of edges while 1 means looking at the whole graph. We tried to test our algorithm in a smaller graph of 3000 edges with different sampling factors.

In the following Table 7, we present the results of this test for two star graphlet.

Table 7: Estimation of 3000 edges, two star

Sampling Factor	Estimated	Relative Error
0.05	46704	0.15
0.10	38124	0.30
0.25	47348	0.14
0.50	40709	0.26
0.75	54841	$8.21 \cdot 10^{-4}$
1	54796	0.00

The general idea is that as the sampling factor increases, the knowledge about the graph also increases, leading to estimate the count closer to the actual number. Table 7 shows that even choosing a very low sampling factor, we had a very low relative error, so we can say that our estimation for the whole graph is reliable.

Table 8: Estimation on real network

Shape	Sampling Factor	Estimated
Triangle	0.005	36400
Two Star	0.005	1094560
Four Clique	0.005	23600
Chordal Cycle	0.005	220160
Tailed Triangle	0.005	2855760
Four Cycle	0.005	78320
Three Star	0.001	31786800
Four Path	0.005	7543960

According to Table 8 the most frequent graphlet in our network is the *three star*, which is reasonable with respect to the structure of our graph; despite the fact that it was calculated considering a lower value for sampling factor for computational limitations.

7 OPEN QUESTION

The results from the link prediction in Section 5 were not satisfying, obtaining only 0.5 in AUC ROC. Considering that our network is made of words, connected based on semantic proximity, we want to analyze whether taking into account the meaning of the words, instead than just meaningless nodes, will increase or not the link prediction values.

To include the meaning of words we use a *word2vec* model. Usually, these models consider the context in which the word is, to calculate its vector, but in our case, we only have independent words so this process is unfeasible. To overcome this problem, we imported a pretrained model based on Google News using KeyedVectors. Each word is represented by a vector of 300 dimensions.

In our dataset, the nodes are composed of single or multiple words. To solve this, we obtained the vectors of each word, computing the average of the vectors if the string was a multi-word expression, and assigning a random vector to unknown words. We built a dataframe based on the existing links (value 1) and a new dataframe having artificial links (value 0) with the same dimension of the first one. We joined them, creating only one dataframe.

Our dataframe had a shape that was not suitable for Logistic Regression. In order to solve this problem, we concatenated both values (node1 and node2) and successively transformed them into a matrix.

Unfortunately, we obtained a mediocre ROC AUC result: 0.48. Since the score did not give us a satisfactory result we tried to improve it via a Neural Network whose parameters are listed in Table 9.

Table 9: Neural Network parameters

Batch size	128
Epochs	100
Learning rate	0.261
Momentum	0.9
Nesterov	False
Activation	Tanh
Regularizer	0.0001
Units	10
Hidden layers	1

As we can see from Figure 19, with NN we got a better result, with a higher ROC AUC Score: 0.79.

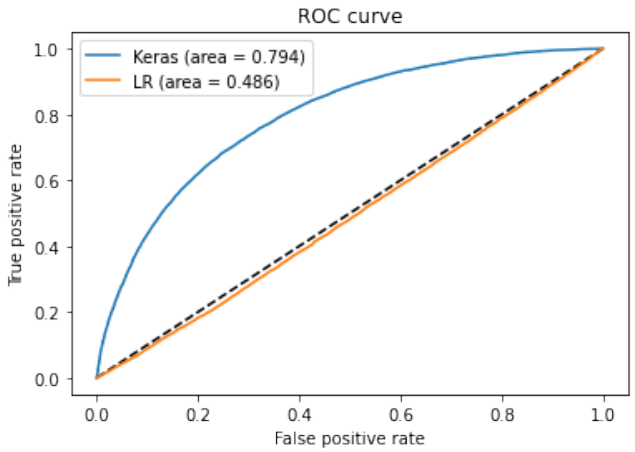

Figure 19: ROC AUC Score

To check whether the new model was able to identify semantic similarity between words, we obtained few words from the predicted nodes, getting the examples showed in Table 10:

Table 10: Examples of link predictions

Node 1	Node 2	Link
Peanut butter cookie	Cookie	1
PrincewaterhouseCooper	Pakistan	1
Internet service provider	Camcorder	1
Makita	Stranger Things	0
Spartanburg	Iphone	0

The predictions labeled as 0 can be considered accurate (between node1 and node2 there is no shared meaning), as for the first label 1 (Peanut butter cookie is clearly correlated with Cookie), but we cannot say that the other two link predictions are correct. In our defense, the used words are sometimes treated as mean between vectors or flagged as unknown.

To additionally evaluate the quality of our model, we set some random words and tested if the link predicted was 0 or 1. Here are a few examples:

- Obama - president: 1
- Obama - cheesecake: 0
- Italy - Naples: 1

Even if our predicted links were not optimal (e.g. PrincewaterhouseCooper and Pakistan), the model can still identify and tag a connection between two words if they are related. We can then conclude that the link prediction based on word2vec, applied on a Semantic Network, gives consistent results.

Future Improvements

An idea to improve the model would be to obtain the context of each word using Google News (or some other platforms) to train the word2vec model directly on real examples, instead of a pretrained model which may not know some tokens, forcing us to use an unknown value. In this manner, the word vectors should be more accurate.

Another attempt would be to create an unbalanced dataframe of links, reflecting the real condition of the graph in which the number of existing links is much less than the number of links that could have been existed.

One last thing that could be interesting to do, is to use node2vec instead of word2vec in the Neural Network since the Logistic Regression result of the former was better with respect to the one of the w2v, so it might be the same with NN as well.

8 DISCUSSION

The project allowed us to better understand the theoretical aspects of complex network analysis, challenging us to apply state-of-the-art algorithms. We used many tools to extract data and create the network for studying the structure and semantic connection of our network, leading to future possible studies in other disciplinary fields such as applications of data science in linguistics, sociology, and computer science.

REFERENCES

[1] A. Grover, J. Leskovec, *Node2vec: Scalable Feature Learning for Networks*, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[2] G. Attardi, *Word Embeddings Tutorial* https://attardi-4.di.unipi.it:8000/hub/user-redirect/lab/tree/HLT/wordEmbeddings.ipynb.

[3] *Graft: An Efficient Graphlet Counting Method for Large Graph Analysis* https://ieeexplore.ieee.org/document/6702435.

[4] G. Rossetti, L. Milli, R. Cazabet. *CDlib: a Python Library to Extract, Compare and Evaluate Communities from Complex Networks* Applied Network Science Journal. 2019. DOI:10.1007/s41109-019-0165-9.

[5] G. Rossetti, L. Milli, S. Rinzivillo, A. Sirbu, D. Pedreschi, F. Giannotti, *NDlib: a Python Library to Model and Analyze Diffusion Processes Over Complex Networks*, Journal of Data Science and Analytics. 2017. DOI:0.1007/s41060-017-0086-6.

[6] P. Joshi, *A Guide to Link Prediction – How to Predict your Future Connections on Facebook* https://www.analyticsvidhya.com/blog/2020/01/link-prediction-how-to-predict-your-future-connections-on-facebook/.