Homework 2 Network Science
A Storm of Nodes

network analysis of the third book of George R.R. Martin's series

Daniele Mari 1231806

January 2020

1 Introduction

In order to perform the analysis required for this second homework, i. e. ranking, community detection and link prediction, the dataset that is used is the one of the first homework. This is because a book series naturally defines communities and, since there are sequels, it is easy to check if the predicted links are correct.

Since many algorithms are non deterministic, a random seed is used in order to guarantee reproducibility, in case one result is highly dependant on the seed it will be specified along the report.

2 Ranking

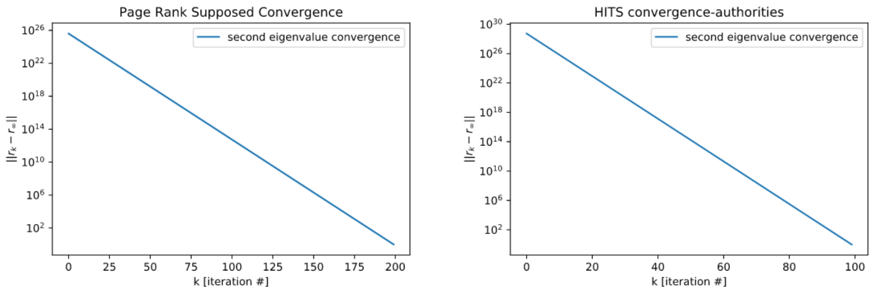
Two of the most known ranking methods in network science are Hits and Pagerank, the following sections discuss the results of the two approaches.

2.1 Pagerank

In this work power iteration is used to calculate Pagerank, since it is possible to find out the error with respect to the linear system solution and it runs order of magnitude faster.

The number of iterations can be chosen by checking when the error, as expected from the second eigenvalue convergence rule, decreases approximately by a factor of 10^{15}. From figure 1a it is possible to see that this happens after more or less 125 iterations.

By running Pagerank on this network it is clear that the result is not very meaningful as the ranking is linearly correlated with the degrees of the nodes (figure 2), this means that Pagerank really doesn't help much in the analysis of the network.



(a) Expected power iteration error convergence for Pagerank

(b) Expected power iteration error convergence for Hits

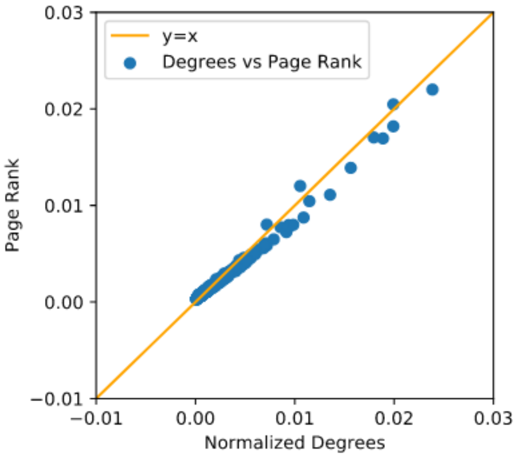


Figure 2: Pagerank against degrees, it is clear that there is a linear relation between the two

.

2.2 Hits

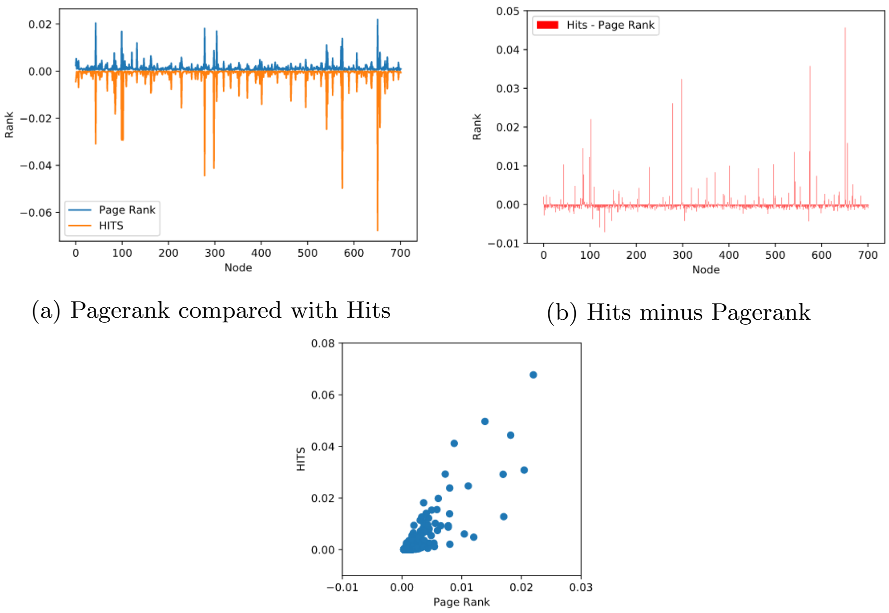
Hits procedure is used to find authorities and hubs, in this case though, since the network is undirected, the two ranks are equivalent.

As for Pagerank also in this case the iterative procedure is used, in figure 1b it is possible to see that the number of iterations needed, in order to decrease the error by a reasonable amount, is more or less 50.

In this particular network using the weighted matrix for this computation leads to better ranking, because it takes into consideration the amount of times two characters interacted.

This time the rank is not proportional to the degrees, meaning that it is also different from Pagerank, this can be clearly seen in figures 3a, 3b, 3c.

By comparing the two rankings, it can be seen from tables 1a and 1b that



(a) Pagerank compared with Hits

(b) Hits minus Pagerank

(c) Pagerank versus Hits

characters that usually interact with other hubs have higher Hits rank than Pagerank.

In particular, if considering hubs, two of the first three characters who improved their position in the rankings the most, are actually the king (Geoffrey Baratheon) and the regent queen (Cersei Lannister). In contrast those whose ranking got worse are actually hubs that are connected to a lot of small nodes. As a matter of fact both Samwell Tarly and Davos Seaworth in the books are very rarely connected with other hubs since they are not famous and live with the common people. This shows how Hits takes into consideration also the fact that a node is connected with other hubs and not only small nodes.

3 Community detection

Community detection in this work is performed with four different approaches, two of them exploit bisection while the other two use clustering algorithms.

3.1 Page Nibble

The first approach is recursive division using Page Nibble; the recursion is stopped whenever the split decreases modularity or the minimum conductance is above a certain threshold. The minimum conductance threshold has to be set because otherwise the recursion would highlight more or less one hundred communities.

Character	Improvement	Page	Hits
Cersei Lannister	10	15	5
Sandor Clegane	8	17	9
Joffrey Baratheon	6	9	3
Sansa Stark	4	5	1
Tywin Lannister	3	11	8
Jaime Lannister	0	2	2
Tyrion Lannister	0	0	0
Robb Stark	0	7	7
Catelyn Stark	-2	4	6
Arya Stark	-3	1	4
Jon Snow	-12	3	15
Sam Tarly	-28	8	36
Davos Seaworth	-37	6	43

Position	HITS	Page
1	Tyrion Lannister	Tyrion Lannister
2	Sansa Stark	Arya Stark
3	Jaime Lannister	Jaime Lannister
4	Joffrey Baratheon	Jon Snow
5	Arya Stark	Catelyn Stark
6	Cersei Lannister	Sansa Stark
7	Catelyn Stark	Davos Seaworth
8	Robb Stark	Robb Stark
9	Tywin Lannister	Sam Tarly
10	Sandor Clegane	Joffrey Baratheon
11	Brienne	Daenerys Targaryen
12	Gregor Clegane	Tywin Lannister
13	Petyr Baelish	Robert Baratheon

Table 1: Hubs overall ranking improvement from Pagerank to Hits (a) and first thirteen ranked characters with both methods(b)

Unfortunately this approach doesn't lead to meaningful community detection in this work, because many communities are really small and contain characters that aren't really related.

This probably happens because hubs are connected with many nodes and this can lead the random walk outside the community very easily.

3.2 Spectral Clustering bisection

The Second approach uses Fiedler's vector, instead of Page nibble, for nodes reordering, all the rest is pretty much the same as the previous technique.

By setting the conductance threshold to 0.25, fifteen communities are highlighted. A couple of these are small communities of fully connected nodes, for example the musicians at the king's wedding that are all listed together in the book.

The remaining communities are actually pretty good as each one of them contains one of the protagonists and the characters with whom they interact the most.

With this approach most of the nodes are in the right community except for the king and his brothers that can be found in their uncle's community which is not too bad but not the best either.

The main problem with this approach is that the minimum conductance value is chosen, manually in order to find a good result, because by only using modularity the algorithm finds more or less one hundred communities.

3.3 K-means

Consider the eigenvectors $v_1, v_2, \dots v_N$ of the normalized Laplacian, relative to the ordered eigenvalues, where N is the number of nodes, and v_2 is Fiedler's vector.

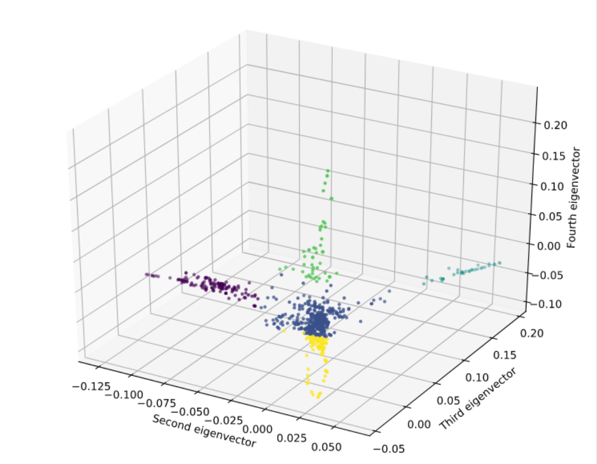
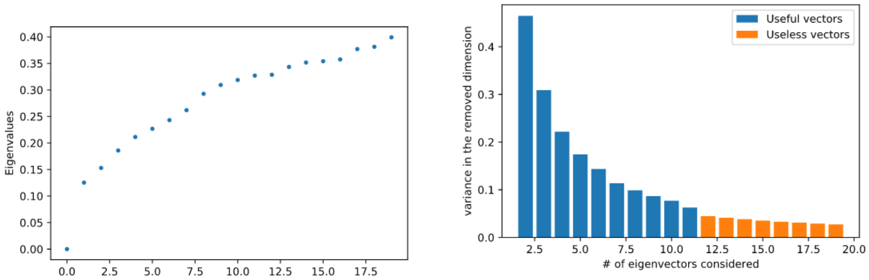


Figure 4: plot of v_1, v_2, v_3 where different colours identify different clusters found by the K-means algorithm



(a) Representation of the 20 smallest eigenvalues

(b) Variance of the last dimension found by reducing by one the dimensionality of the space by using PCA

Spectral clustering only uses Fiedler's vector to compute the communities, but also $v_3, \dots v_N$ can be used, in particular by assigning to each node n_i a low dimensional representation $[v_{2,i}, \dots v_{k,i}]$ it might be possible to cluster the nodes.

The value k is usually chosen as the number of eigenvectors between Fiedler's vector and the eigengap. In this case the eigengap is between the first and the second smallest eigenvalues (figure 5a) so theoretically the eigenvectors after Fiedler's shouldn't be very meaningful. Even so by plotting v_2, v_3, v_4 in figure 4 it is possible to see the emergence of some meaningful communities. Because of that another approach is taken in order to choose the best value of k.

In general, adding more coordinates to the node representation becomes useless when the new eigenvector doesn't add information to the current representation. This means that even if the dimensionality of the space increases, the data still lies on a lower dimensional space. By using PCA(Principal Component Analysis) it is possible to reduce the dimensionality of a vector space and to find out the variance of the data in the dimensions that were removed. By adding one dimension at the time and by reducing by one the dimension of the space with PCA it is possible to see that the variance of the last dimension decreases as the the number of eigenvectors increases with an exponential trend. From figure 1a it is clear that at one point the variance is reduced more than expected, this is, probably, where the next eigenvectors start to be useless for community detection.

At this point k is known, the optimal number of clusters can be found by selecting the one that computes the split with the highest modularity, a plot of the modularity variation as the number of clusters increases can be seen in figure 6a.

Compared to bisection this method does not require to manually tweak values meaning that it is more robust, the communities that it finds are more general and not just the groups of people that interact with the main characters, in this case the clusters tend to contain people that live in the same region.

The first problem with this approach is that small fully connected communities are found, this though is a problem that depends on how the network was generated, as nodes that are named nearby in the text are connected even if they are not related so it is not easy to fix it.

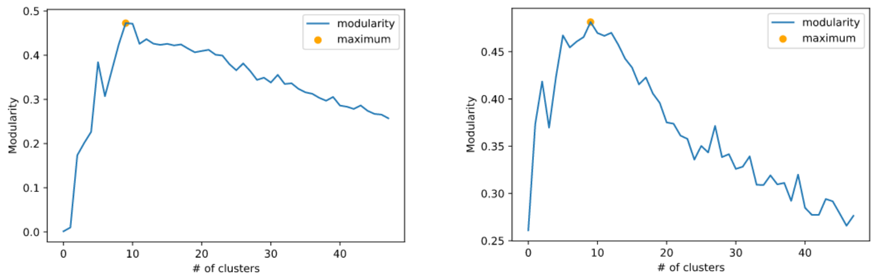
The second problem consists in the fact that a couple of characters are not assigned to the right community but to a suboptimal one. For example Stannis Baratheon can be found in the same community as his relatives even if he lives far away and doesn't really interact with them. This is probably due to the fact that K-means can only find spherical clusters but from the 3D representation it is clear that some are actually elliptical.

3.4 Gaussian Mixtures

In order to fix the spherical clusters problem gaussian mixtures can be used instead of K-means. This method can find elliptical clusters since it computes the gaussians that better fit the data and assigns each point to a community by using maximum likelihood. The number of clusters is still chosen by selecting the one that maximizes modularity (figure 6b)

One nice feature of the gaussian mixtures is that by maximum likelihood estimation it can assign to each node a communities probability distribution, making it an algorithm for basic overlapping communities. For example in this case the character Jaime Lannister has more or less 60% probability of being in the Lannisters community, which is correct, and 40% of being in the Arya Stark community that is suboptimal because he interacted a lot with characters in that community.

This method is the one that works better for this network, because it finds general communities and all hubs and secondary nodes are classified correctly.



(a) Modularity value for K-means as the number of clusters increases

(b) Modularity value for Gaussian Mixtures as the number of clusters increases

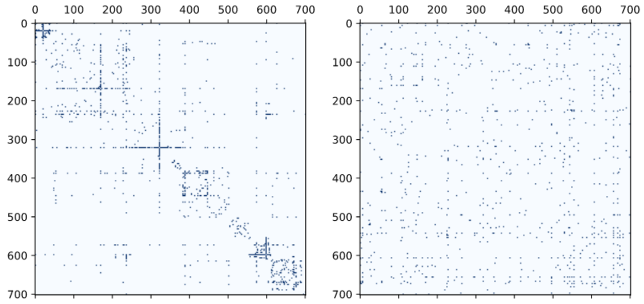


Figure 7: Adjacency matrix with and without community highlighting

It is hard to check if all characters are in the correct community since they are too many and it isn't easy to remember who they all are.

Also this time some small communities with fully connected nodes emerge.

Two representations of the communities found with Gaussian Mixtures can be seen in figures 7, 8.

3.5 Armies Robustness

Six of the communities found by Gaussian Mixtures are actually good representations of armies appearing in the book, since they contain all characters involved in the chain of command. In general we can say that when the chain of command breaks then the faction has lost the war.

It might be possible to simulate the strength of an army by calculating how it responds to random node removals, that represents random deaths in battle, and to hubs removal that can be seen as thoughtful attacks like the murder of the king and his most important councilors. The results of random node and hubs removal can be seen in figure 9.

In this case choosing the breaking point by using the inhomogeneity ratio is not meaningful because the networks are too small, so instead if the biggest component in the army is smaller than the 5% of it's original dimension then

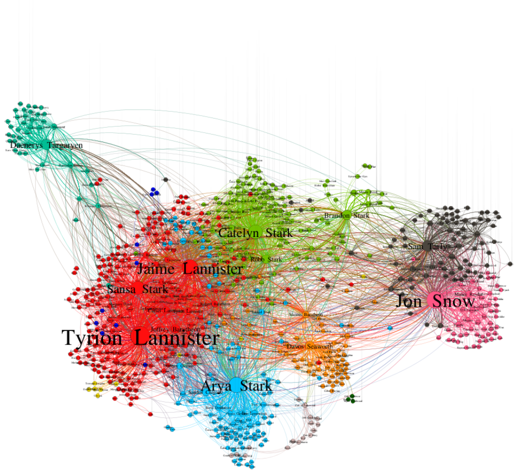


Figure 8: Communities representation plotted with Gephy

the army is considered as defeated.

The most important war in this book is the one between the Lannisters and Robb's army. The latter didn't loose a single battle but was defeated because the General of the Lannister convinced one of Robb's allies to murder him and all the most important characters in his army, i.e the hubs. By analyzing the robustness of the two armies, it is clear that they have more or less the same robustness to random attacks, but, considering the fact that Robb was winning every battle, he would probably have won the war. This means that the only way for the Lannisters to win was very likely to perform hubs removal, that is exactly what they did.

It is also very interesting to see how these two armies are the ones with the smallest resistance to hubs removal. This might be due to the fact that both the armies strongly rely on the guidance of their lords, so if they die the army looses its purpose.

In contrast the army of the free folks has higher resistance to hubs removal because the soldiers of this army have only a symbolic leader and are escaping a greater enemy meaning that even if the hubs fall, the others would still keep fighting.

This method is far from perfect, for example Stannis's army has very high resistance to hubs removal just because of the threshold value, since, as it can be seen, the dimension of the GC becomes very small really fast and then stabilizes.

For the remaining armies it is difficult to find an interpretation for the breaking values.

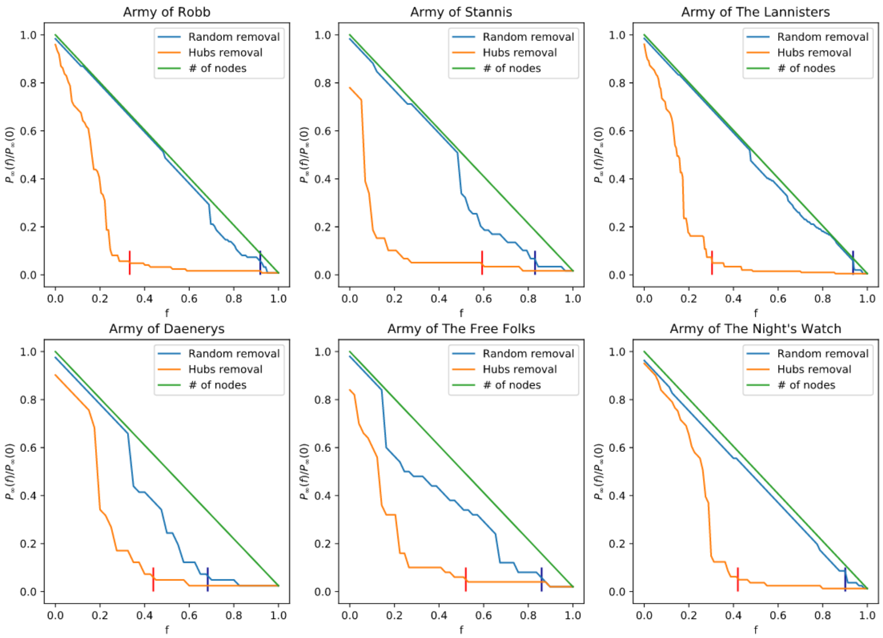


Figure 9: Robustness of the armies in the book

4 Link Prediction

For what concerns link prediction many algorithm were implemented and the best one is chosen by comparing their performance with the AUC_ROC metric and the precision metric. The implemented algorithms are: Common neighbours, Adamic Adar, Resource Allocation, Katz, Random walk with restart, Local random walk and Superposed random walk

In Katz the similarity formula is

$$S_{katz} = \sum_{l \geq 1} \beta^l A^l$$

but as l increases the factors have less and less relevance because of β and the computation becomes more expensive because the sparsity of the matrix disappears. So a maximum value for l can be chosen leading to the approximate formula

$$S_{katz} = \sum_{l=1}^{l_{max}} \beta^l A^l$$

The optimal values of l_{max} and β can be found by performing a grid search, the results can be seen in figure 10. These parameters though are highly dependant on the random seed, but in general with this approach the overall performance of the algorithm doesn't change.

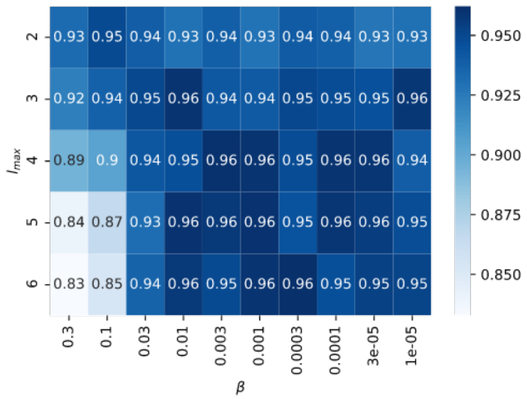
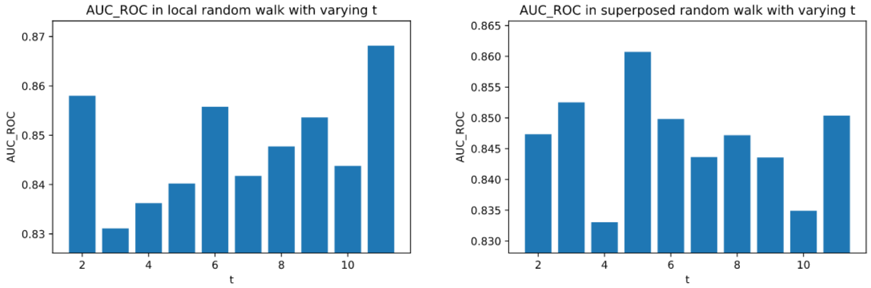


Figure 10: Grid search results from Katz



(a) AUC_ROC for different values of l in local random walk

(b) AUC_ROC for different values of l_{max} in Superposed random walk

For Local random walk and Superposed random walk a similar procedure has to be followed in order to find the best value of l, the results can be seen from figures 11a, 11b.

The precisions for these three algorithms are computed using the best parameters for AUC_ROC.

The final scores for each algorithm, in table 2, show that the most performing algorithm for this network is Resource allocation, while the second best is Random walk with restart. These results are independent of the seed as in all the tests these two are always the best ones with Resource allocation far ahead with the precision score.

Links predicted with random walk with restart mainly involve secondary characters so they are not really appealing, in contrast resource allocation makes very interesting predictions. Table 3 presents the ten most likely links according to resource allocation, most of these are actually links that form in one of the sequels, some of them involve a dead character so the two nodes won't ever link and some others were already active in previous books.

It is very nice to see that the first three links would actually be major spoilers

Algorithm	AUC_ROC	Precision
Common neighbours	0.938389	0.197216
Adamic Adar	0.964562	0.243619
Resource Allocation	0.975685	0.385151
Katz	0.938958	0.197216
Random walk with restart	0.975374	0.292343
Local random walk	0.840872	0.071926
Superposed random walk	0.829353	0.058005

Table 2: Table of the scores of different link prediction algorithms

Character1	Character2	Note
Daenerys Targaryen	Tyrion Lannister	during fifth book
Davos Seaworth	Samwell Tarly	during fourth book
Daenerys Targaryen	Arya Stark	during eight season
Catelyn Stark	Samwell Tarly	Catelyn is dead
Jayme Lannister	Lothar Frey	during fourth book
Gregor Clegane	Sansa Stark	Gregor is dead, already met in first book
Brienne	Tywin Lannister	Tywin is dead
Devan	Stannis Baratheon	Devan is Stannis's squire
Kevan Lannister	Robb Stark	Robb is dead
Hosteen Frey	Raymund Frey	they are brothers

Table 3: Ten most likely links according to Resource Allocation

for a reader that just finished the book.

It is interesting to check what are the most likely prediction involving some characters that had no new link in the first ten, in this case Jon Snow, Roose Bolton and Cersei Lannister.

Jon Snow's most likely link is particularly surprising because in the books it is not known, yet, who his real parents are, but the algorithm predicted that he would link with Aegon Targaryen that, according to the tv series, is actually one of his ancestors and by coincidence is actually his real name.

Roose Bolton connects with Sansa Stark and the two actually meet in season five, Cersei's link is with Roose but unfortunately the two haven't met neither in the tv series nor in the books until now.

5 Conclusions

The proposed algorithms work quite well since the results that are found coincide with what was expected at the beginning of this work.

The communities and the predicted links are correct and the obtained results actually enforce some of the choices made by the author in the book.

Also through community detection it is clear that there are many small clusters of fully connected nodes that are due to how the network was built.