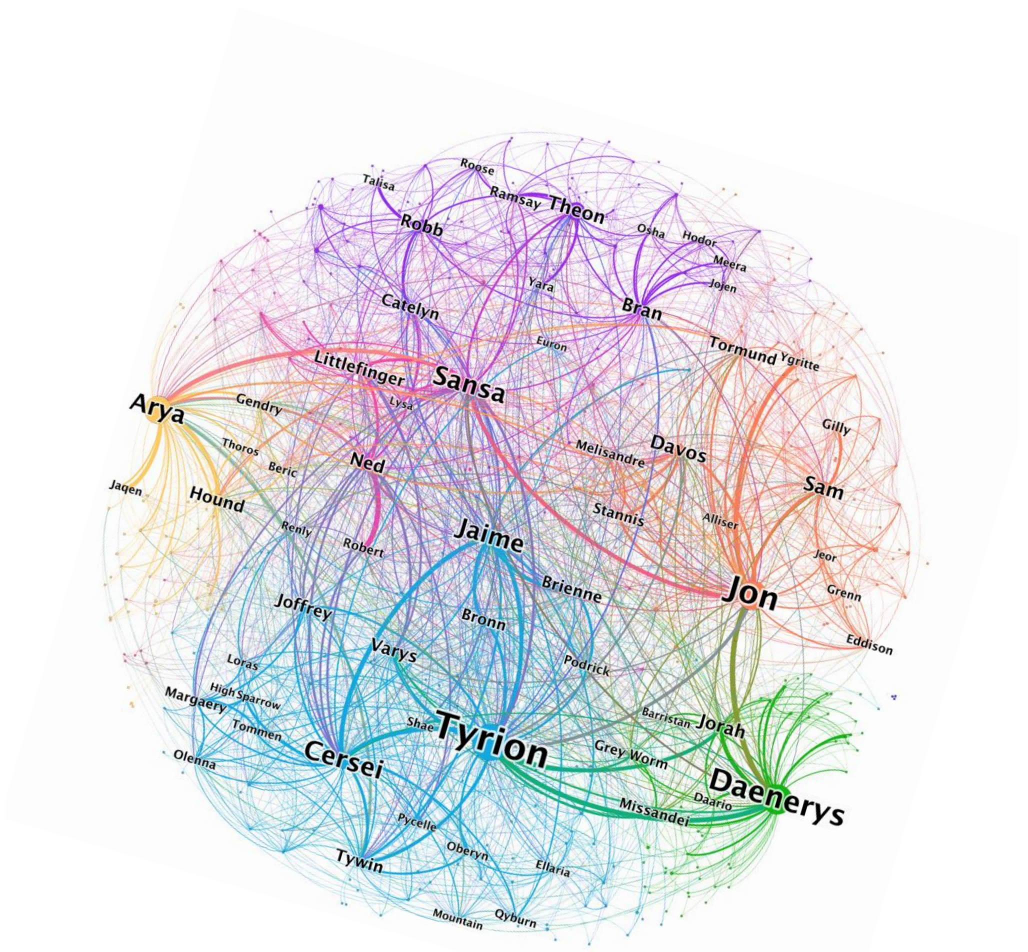


Network Analysis of Game of Thrones

Datacamp Project

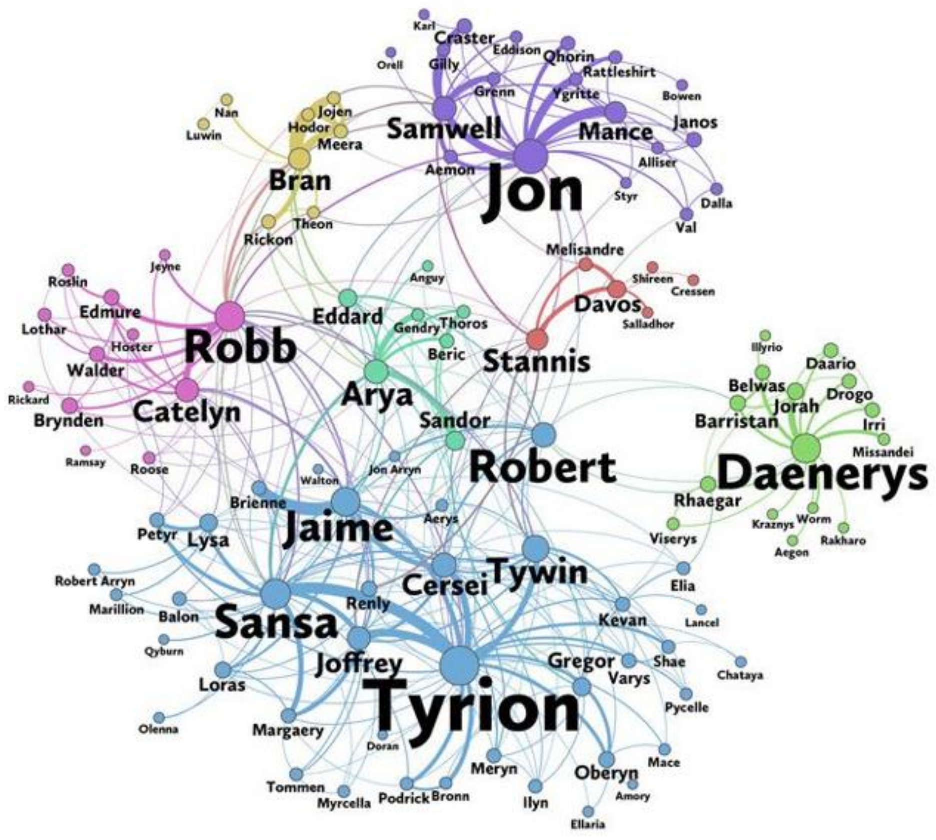
Shreya Gopal Sundari

Contents

INTRO: This project is to analyze the network of characters in Game of Thrones and how it changes over the course of the books.

1. Project Description

Jon Snow, Daenerys Targaryen, or Tyrion Lannister? Who is the most important character in Game of Thrones? Let's see what mathematics can tell us about this!



In this project, we will look at the character co-occurrence network and its evolution over the five books in R.R. Martin's hugely popular book series A Song of Ice and Fire (perhaps better known as the TV show Game of Thrones). And also look at how the importance of the characters changes over the books using different centrality measures.

1.1. Requirements

For this project, you should be familiar with the

- networkx package

- different network centrality measures
- pandas package
- manipulate DataFrames

2. Dataset Details

- This project uses a dataset parsed by Andrew J. Beveridge and Jie Shan which is available here.
- For more information on this dataset have a look at the Network of Thrones blog.

3. Tasks, Instructions & Results

The Jupyter notebook is predefined and has information on the tasks to complete this project. The project consists of nine tasks which mentioned below along with the instructions to execute them:

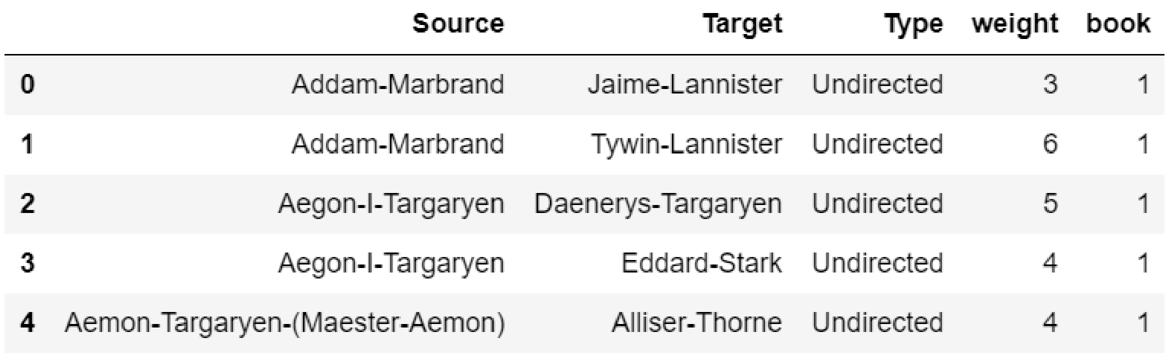
1. Winter is Coming. Let's load the dataset ASAP!
2. Time for some Network of Thrones
3. Populate the network with the DataFrame
4. The most important character in Game of Thrones
5. The evolution of character importance
6. What's up with Stannis Baratheon?
7. What does Google PageRank tell us about GoT?
8. Correlation between different measures
9. Conclusion

3.1. Task 1: Winter is Coming. Let's load the dataset ASAP!

Load in and inspect the edge list of the first book.

- Import the `pandas` module.
- Load the csv file for book 1 from `"datasets/book1.csv"` and assign it to `book1`.
- Print out the `head` (first 5 rows by default) of the DataFrame `book1`

The result of this task is as follows:



	Source	Target	Type	weight	book
0	Addam-Marbrand	Jaime-Lannister	Undirected	3	1
1	Addam-Marbrand	Tywin-Lannister	Undirected	6	1
2	Aegon-I-Targaryen	Daenerys-Targaryen	Undirected	5	1
3	Aegon-I-Targaryen	Eddard-Stark	Undirected	4	1
4	Aemon-Targaryen-(Maester-Aemon)	Alliser-Thorne	Undirected	4	1

Figure: Book1 DataFrame

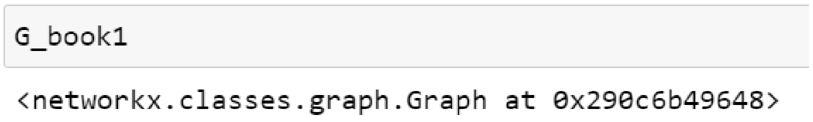
3.2. Task 2: Time for some Network of Thrones

Create a graph object for the first book.

- Import the networkx module and give it the alias nx.
- Create an empty Graph object and assign it to the variable G_book1.

networkx provides different kind of graph objects, graph, digraph, multigraph, multidigraph. In this case, you will create a graph because the network is undirected, that is, an edge from character A to character B implies that there exists an edge the other way too, from character B to character A.

The result of this task is to create a graph object as follows:



```
G_book1
```

```
<networkx.classes.graph.Graph at 0x290c6b49648>
```

Figure: Graph Object Created

3.3 Task 3: Populate the network with the DataFrame

Add nodes and edges information to the network for book 1.

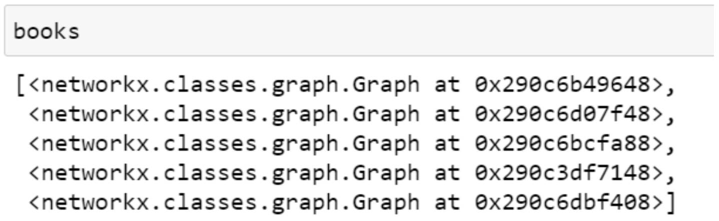
- Iterate through the DataFrame book1 row-wise using iterrows().
- Add edges to the graph object G_book1 using add_edge().

To populate the graph you need to iterate through book1 to add weighted edges to the G_book1 network using add_edge(source, target, weight=). Remember that the weight= argument needs to be named explicitly when using add_edge. When iterating through the DataFrame, keep in mind that iterrows() returns a 2-tuple of an index and a pandas Series object. You need to use only the second part, that is, the pandas Series object to get all the information needed to populate the network.

You only need the source, target, and weight column to create the character co-occurrence network so only add this information to G_book1.

Since we want to analyze all five books, we have supplied code for creating the graphs for the other four books.

The result of this task is to create a list containing graph objects for all 5 books as follows:



```
books
```

```
[<networkx.classes.graph.Graph at 0x290c6b49648>,
 <networkx.classes.graph.Graph at 0x290c6d07f48>,
 <networkx.classes.graph.Graph at 0x290c6bcfa88>,
 <networkx.classes.graph.Graph at 0x290c3df7148>,
 <networkx.classes.graph.Graph at 0x290c6dbf408>]
```

Figure: 5 Graph Objects with Data

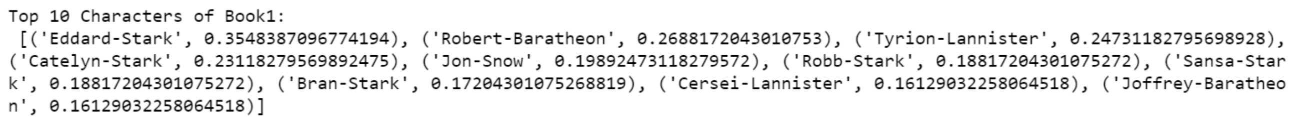
3.4 Task 4: The most important character in Game of Thrones

Find the most important characters according to degree centrality.

- Use nx.degree_centrality(graph) to calculate the centrality of all nodes from the first book (book[0]) and the fifth book (book[4]).
- Sort the resulting dictionaries deg_cen_book1 and deg_cen_book5 according to decreasing values and store the top 10 in sorted_deg_cen_book1 and sorted_deg_cen_book5.
- Print out sorted_deg_cen_book1 and sorted_deg_cen_book5.

To calculate degree centrality every node is assigned a number between 0 and 1 by computing the degree (the number of neighbors) and normalizing it by the total possible number of neighbors it can have, i.e n - 1 where n is the number of nodes in the network.

The result of this task is as follows:



```
Top 10 Characters of Book1:
 [('Eddard-Stark', 0.3548387096774194), ('Robert-Baratheon', 0.2688172043010753), ('Tyrion-Lannister', 0.24731182795698928),
('Catelyn-Stark', 0.23118279569892475), ('Jon-Snow', 0.19892473118279572), ('Robb-Stark', 0.18817204301075272), ('Sansa-Star
k', 0.18817204301075272), ('Bran-Stark', 0.17204301075268819), ('Cersei-Lannister', 0.16129032258064518), ('Joffrey-Baratheo
n', 0.16129032258064518)]
```

Figure: Important Characters of GOT Book1

```
Top 10 Characters of Book5:
 [('Jon-Snow', 0.1962025316455696), ('Daenerys-Targaryen', 0.18354430379746836), ('Stannis-Baratheon', 0.14873417721518986),
('Tyrion-Lannister', 0.10443037974683544), ('Theon-Greyjoy', 0.10443037974683544), ('Cersei-Lannister', 0.0886075949367088
6), ('Barristan-Selmy', 0.07911392405063292), ('Hizdahr-zo-Loraq', 0.06962025316455696), ('Asha-Greyjoy', 0.0569620253164556
94), ('Melisandre', 0.05379746835443038)]
```

Figure: Important Characters of GOT Book 5

3.5. Task 5: The evolution of character importance

Plot the evolution of degree centrality over the books for some of the characters.

- You are given a list `evol` that contains the computed degree centrality from all the books.
- Create a DataFrame with character names as columns, and index as books, where every entry is the degree centrality of the character in that particular book using `pd.DataFrame.from_records`.
- Plot the columns `Eddard-Stark`, `Tyrion-Lannister`, `Jon-Snow` from the DataFrame `degree_evol_df` using `.plot()`.

The resulting plot is as follows:

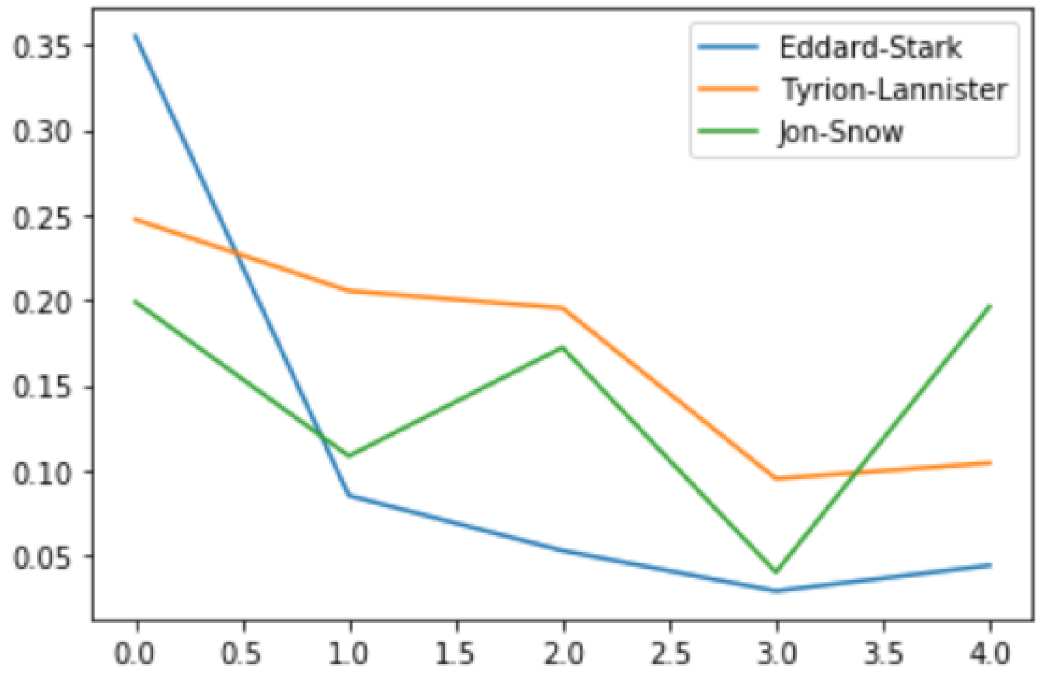


Figure: Evolution of Eddard Stark, Jon Snow, and Tyrion

3.6. Task 6: What's up with Stannis Baratheon?

Find the importance and evolution of characters according to betweenness centrality.

- Use `nx.betweenness_centrality(graph, weight='weight')` to calculate the weighted betweenness centrality of all the books.
- Create a DataFrame `betweenness_evol_df` just like in the previous task (`degree_evol_df`) but this time do this for betweenness centrality.
- You have been given code that finds the top 4 characters in each book and puts them into `list_of_char`.
- Plot the columns in `list_of_char` in the DataFrame `betweenness_evol_df` using `.plot()` with the argument `figsize=(13, 7)`.

Remember to use `.fillna(0)` when creating `betweenness_evol_df` as it's possible that a character is not in every book. This would result in `NaN` entries and we want to avoid that so we replace `NaN` with zero.

The intuition behind betweenness centrality is to find nodes which hold the network together, that is, if you remove such a node you are breaking apart the network. In a more mathematical way, betweenness centrality is calculated by finding shortest paths between all pairs of nodes and finding the node through which most of the paths pass.

The resulting plot obtained in this task is as follows:

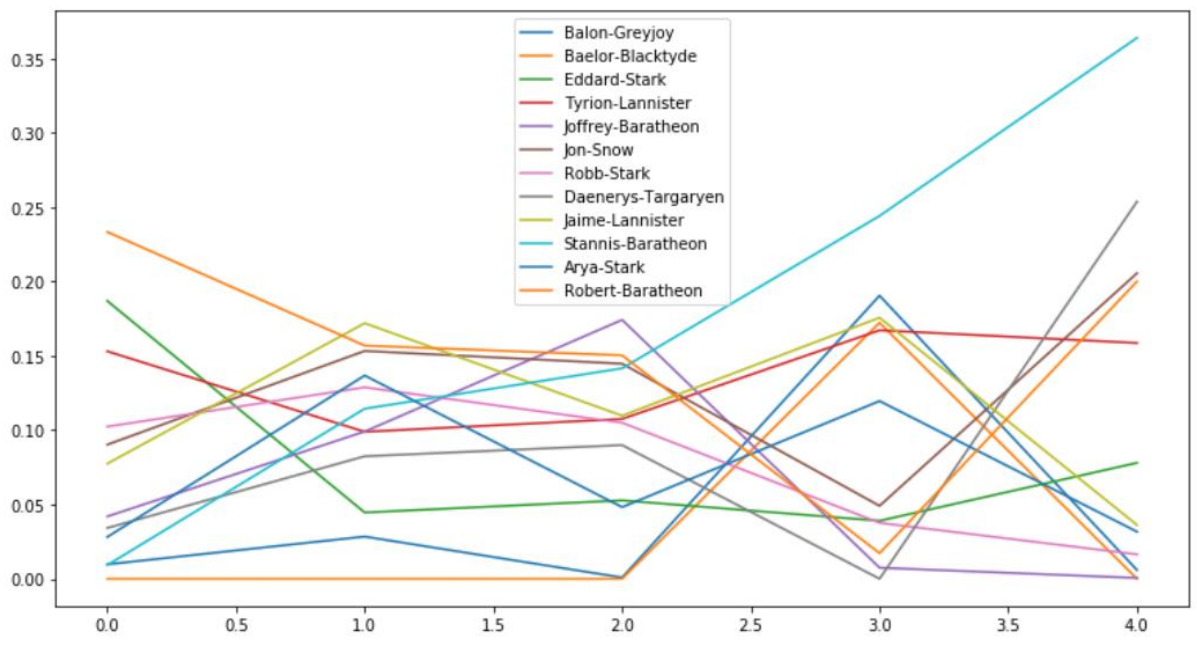


Figure: Evolution of Top 4 characters of every book - *betweenness centrality*

3.7. Task 7: What does Google PageRank tell us about GoT?

Find the importance and evolution of characters according to PageRank.

- Repeat the previous task for PageRank using `nx.pagerank()` and create a list of PageRank measures for all the books.
- Create a DataFrame using `pd.DataFrame.from_records`.
- Plot the columns in `list_of_char` in the DataFrame `pagerank_evol_df` using `.plot()` with the argument `figsize=(13, 7)`.

The resulting plot obtained in this task is as follows:

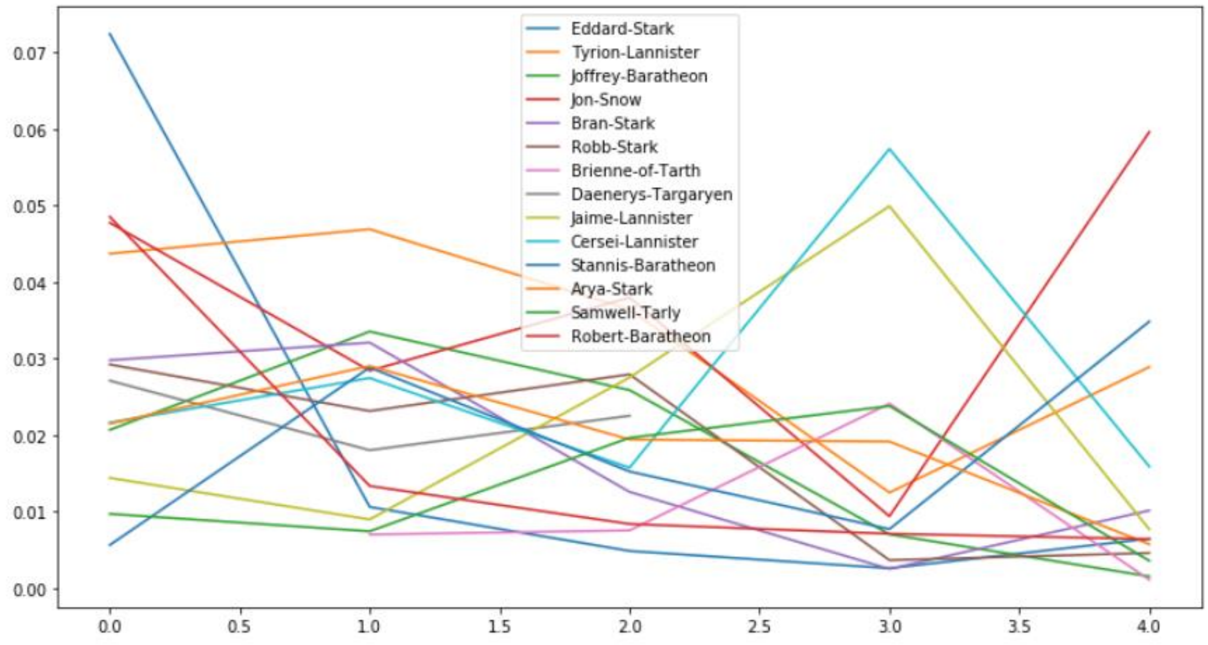


Figure: Evolution of Top 4 characters of every book - P*ageRank*

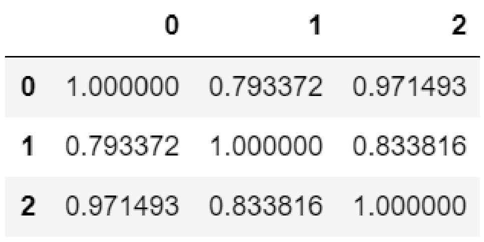
3.8. Task 8: Correlation between different measures

Find the correlation between the three methods of measuring importance.

- Create a DataFrame using `pd.DataFrame.from_records` using the list of all `measures` for `books[4]`, and assign it to `cor`.
- Calculate the correlation using `.corr()`.

Make sure to take the transpose of the DataFrame `cor`, that is, `cor.T` first before calculating the correlation otherwise you get the correlation between characters and not the correlation between measures.

The obtained correlation between PageRank, betweenness centrality and degree centrality for the fifth book using Pearson correlation is as follows:



	0	1	2
0	1.000000	0.793372	0.971493
1	0.793372	1.000000	0.833816
2	0.971493	0.833816	1.000000

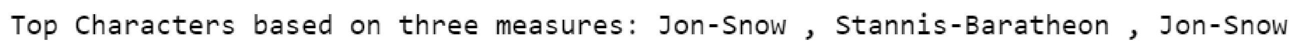
Figure: Correlation between different measures

3.9. Task 9: Conclusion

- Use the `cor` DataFrame to find the *most* important character in the fifth book according to degree centrality, betweenness centrality, and PageRank.
- Print out the top character(s) according to these three measures.

This task can easily be completed using the **idxmax method**.

The resulted top characters are as follows:



```
Top Characters based on three measures: Jon-Snow , Stannis-Baratheon , Jon-Snow
```

Figure: Top Characters

4. Further Reading

If you more interested in network analysis you can have a look at the following resources:

- **networkofthrones** is a blog dedicated to data collection and all things networks about Game of Thrones.
- **bookworm**: Extracting social networks from novels.

5. References

[1] https://learn.datacamp.com/projects/76

[2] https://networkx.github.io/documentation/stable/

[3] https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas

[4] https://networkx.github.io/documentation/stable/reference/classes/generated/networkx.Graph.add_edge.html

[5] https://stackoverflow.com/questions/613183/how-do-i-sort-a-dictionary-by-value/2258273#2258273

[6] https://pandas.pydata.org/pandas- docs/stable/reference/api/pandas.DataFrame.from_records.html

[7] https://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.idxmax.html