Ram Gautam
Data Science
Programming Assignment 2

Prediction of Character's Survival in Game of Thrones

Introduction

The task of the project is to predict the characters those are likely to survive at the end of popular television series, Game of Thrones (GOT). The five text books, and a .cbor (got.cbor) is given as the source of information. The got.cbor was used to extract the features, and then those features were employed in machine learning algorithms to classify either the character is dead or alive in the Game of Thrones. F1 score is used as the evaluation method to check the accuracy of the Machine Learning Methods.

Feature Extraction Methods

The major features that were extracted to train the Machine Learning algorithms are dead_alive, gender, nobility, and degree of Centrality. The got.cbor file was indexed using lucene indexing, and content of each page were extracted using lucene searcher. The idea was to search through the all the pages that mentions the character's name. At this stage, all the pages that mentions either the first name or the full name of the character is linked, and all the lines of the pages were carefully examined to extract the following features.

1. **Dead or Alive:** In this case, each sections of the pages are considered that mentions the character's name. The TFIDF score for all the death defining words (die, died, death, and so on) is calculated along with the TFIDF score for all the life (alive, survive, victory) defining words from that page. Now, the life or death of that character is predicted based on the TFIDFscore. If the TFIDF score is for death is higher than the that of life, then the character will die. A list of stop words (a, an, the, and so on) were defined, and were excluded while computing the TFIDF score.

2. **Nobility:** The character is linked to the pages that mentions the character name, and TFIDF score is used to define the either the character is from the Noble house or not. The TFIDF score for the all Noble houses such as House Stark, House Lannister, and so on is computed if there is any page link for that character. If the TFIDF score for that character is greater than 0, then the Nobility for that character is set to 1 (meaning the is Noble).

3. **Gender:** The TFIDF score is used to determine either the character is male or female. Initially, all the sections of the pages that mentions character is collected, and the TFIDF score of male and female defining words is computed. If TFIDF score for female is higher than that of male, the character is supposed to be female, and vice versa.

4. **Degree of Centrality:** The Degree of Centrality is the measure of how many connections or links the character has. The character with higher centrality measure is supposed to live longer than the character who has no connection at all. The connection (node and edge) for each character was stored into the list, and a network created using python Networknx as shown in Fig1. The node and edge were collected from the sections of pages that mentions the character's name.

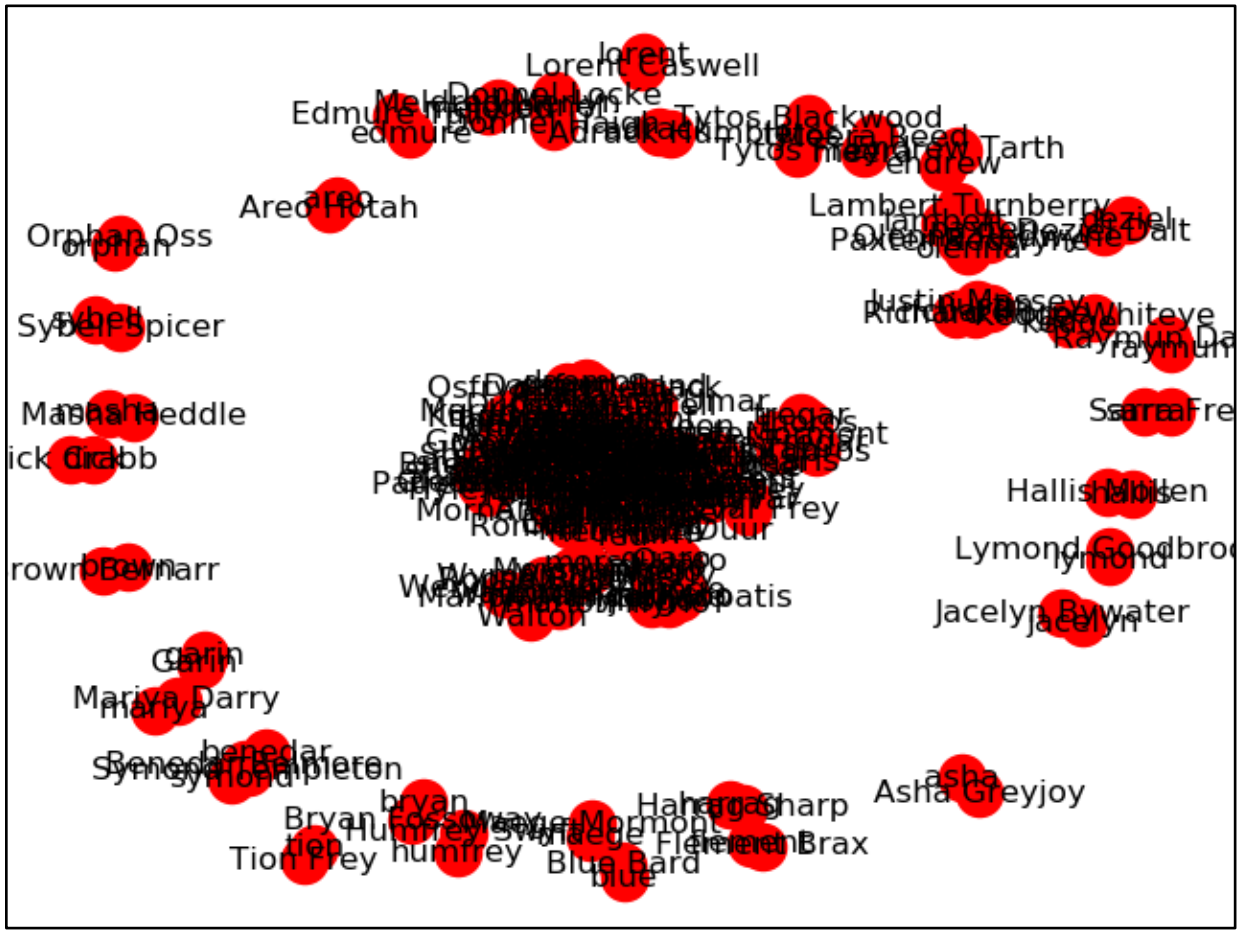

Fig1: GOT Network of Characters

Results and Discussion

Once the features were extracted, the machine learning methods were used to classify either the character is dead or alive. The machine learning methods used in the project are Logistic Regression, Support Vector Machine, Naïve Bayes, and Decision Tree. The original list of about 1000 characters was split into 10 subsets each containing 200 characters, and F1 scores for all those subsets are tabulated as shown in Table 01.

Methods	Logistic Regression	SVM	Naïve Bayes	Decision Tree	mean	std
Subset0	0.4794	0.4884	0.4884	0.4884	0.4861	0.0045
Subset1	0.2622	0.2622	0.2672	0.2692	0.2652	0.0035
Subset2	0.1873	0.1916	0.1916	0.1916	0.1905	0.0021
Subset3	0.2205	0.2205	0.2205	0.2205	0.2205	0.0000
Subset4	0.2366	0.2353	0.2353	0.2353	0.2356	0.0006
Subset5	0.2113	0.2179	0.2179	0.2179	0.2162	0.0033
Subset6	0.2375	0.2335	0.2335	0.2335	0.2345	0.0020
Subset7	0.2520	0.2520	0.2520	0.2520	0.2520	0.0000
Subset8	0.1961	0.1962	0.1961	0.1961	0.1961	0.0001
Subset9	0.2520	0.2520	0.2520	0.2520	0.2520	0.0000
mean	0.2534	0.2549	0.2554	0.2556		
std	0.0831	0.0852	0.0853	0.0853		
std error	0.0262	0.0269	0.0269	0.0269		

Table 01: F1 Score for all the Machine Learning Methods for different subsets

All the methods performed about the same based on the mean and standard error across all the subsets. I could not observer any two methods that were not overlapped (meanA - stderrA > meanB + stderrB was never found where A and B are any two methods). However, the mean F1 score of Logistic Regression was found to be 0.2534 with an standard error of 0.0262, the mean F1 score for SVM was found to be 0.2549 with an standard error of 0.0269, the mean F1 score for Naïve Bayes was found to be 0.2554 with an standard error of 0.0269, and the mean F1 score for Decision Trees was found to be 0.2556 with an standard error of 0.0269. The Decision Tree has the highest F1 score.

I also looked how each subset performed across all the machine learning methods. I observed that all the machine learning methods got highest F1 score on Subset0, and all the machine learning methods performed poor on the Subset2. An error bar was plotted to show the performance of each machine learning method as shown in Fig. 3 below.

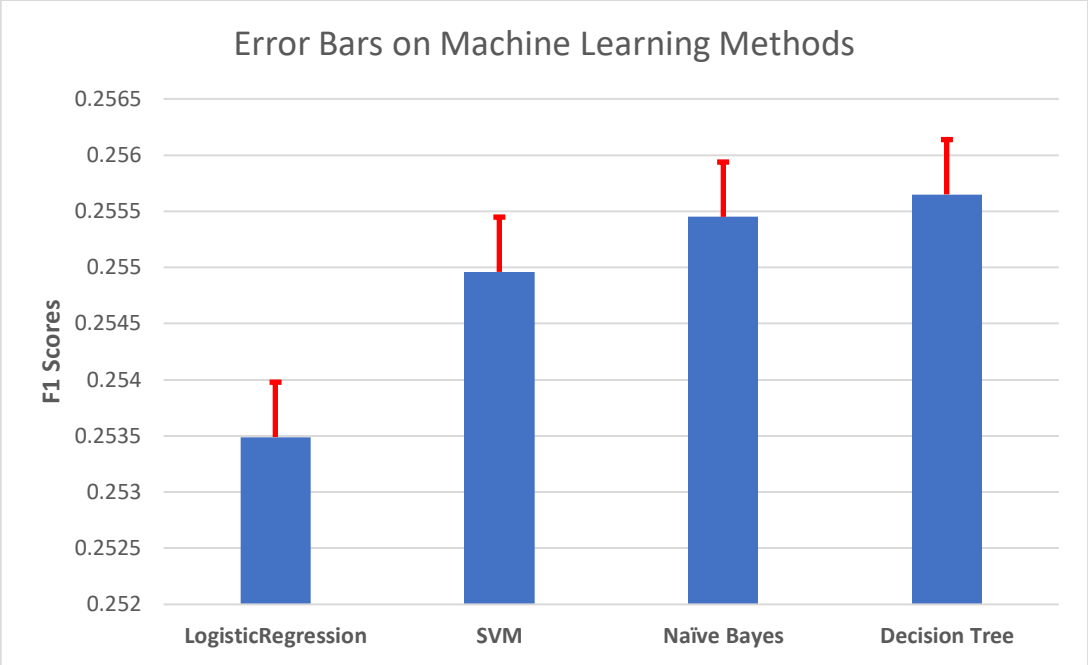

Fig 03: Error Bars on Subsets

Conclusion

I was not able to come to a clear conclusion which machine learning method would be the best over the other machine learning method though the F1 score of SVM, Naïve Bayes, and Decision Tree is the highest. Based on the aforementioned Table 01, Subset0 worked very well for all the machine learning methods. The F1 scores were computed using trec_eval on the run files and predefined death.qrel file.

I did not work much in ranking the entity. Going forward, I would like work more on entity ranking, and find better entity mention for the character. I looked each page using the first name, and full name which was not a decent trick though more than enough pages were retrieved just using the first name and last name.

It was an exciting project and I used the concepts of entity linking, entity searching, and entity network to extract the features. The features were evaluated using the machine learning methods over the 10 subsets of characters.

Run Instructions

```
# Team8

**Run Instructions**

1. Install Maven
2. Pull code into your Machine
   https://gitlab.cs.unh.edu/cs953-spring-2020/team8.git
3. cd Submission2
4. chmod 777 runScript.sh
5. ./runScript.sh

**Python**
1. Install Python 3
   sudo apt-get install python3
2. Install networkx:
   pip install networkx

**Run run Files only **
#NOTE: Running code can be a bit time consuming. Please follow following instructions
#      if you just want to see the results

1. cd Submission2
2. chmod 777 run_Results_From_runFiles.sh
```

An example of run is shown below:

```
[rg1088@dsci-g1 team8]$ vim run_Results_From_runFiles.sh
[rg1088@dsci-g1 team8]$ ./run_Results_From_runFiles.sh
--- F1 SCORE: LOGISTIC REGRESSION START ---
set_F                   all     0.4794
set_F                   all     0.2622
set_F                   all     0.1873
set_F                   all     0.2205
set_F                   all     0.2366
set_F                   all     0.2113
set_F                   all     0.2375
set_F                   all     0.2520
set_F                   all     0.1961
set_F                   all     0.2520
--- F1 SCORE: LOGISTIC REGRESSION END ---
--- F1 SCORE: SVM START ---
set_F                   all     0.4884
set_F                   all     0.2622
set_F                   all     0.1916
set_F                   all     0.2205
set_F                   all     0.2353
set_F                   all     0.2179
set_F                   all     0.2335
set_F                   all     0.2520
set_F                   all     0.1961
set_F                   all     0.2520
--- F1 SCORE: SVM END ---
--- F1 SCORE: GaussianNaiveBayes START ---
set_F                   all     0.4884
set_F                   all     0.2672
set_F                   all     0.1916
set_F                   all     0.2205
set_F                   all     0.2353
set_F                   all     0.2179
set_F                   all     0.2335
set_F                   all     0.2520
set_F                   all     0.1961
set_F                   all     0.2520
--- F1 SCORE: GaussioanNaiveBayes END ---
--- F1 SCORE: Decision Tree START ---
set_F                   all     0.4884
set_F                   all     0.2692
set_F                   all     0.1916
set_F                   all     0.2205
set_F                   all     0.2353
set_F                   all     0.2179
set_F                   all     0.2335
set_F                   all     0.2520
set_F                   all     0.1961
set_F                   all     0.2520
--- F1 SCORE: Decision Tree END ---
[rg1088@dsci-g1 team8]$
```

Reference

1. https://gist.github.com/guenodz/d5add59b31114a3a3c66
   This concept was improvised to compute the TFIDF score of each term.