Submission 3 Report- Team 3

Abhinav Gupta[*]
ag1226@wildcats.unh.edu
University of New Hampshire
Durham, New Hampshire

Stavan Anjaria[*]
spa1019@wildcats.unh.edu
University of New Hampshire
Durham, New Hampshire

Gayathri Venkatasrinivasan[*]
gv1020@wildcats.unh.edu
University of New Hampshire
Durham, New Hampshire

1 ABSTRACT

The prediction of whether a character will be dead or alive, can be approached using different text classification methods. With the wikipedia and the books as the dataset, our idea for this submission is to develop some new features we have not explored earlier by using sentiment analysis and improve our entity linker, later feed them into Machine Learning methods. We plan to have more machine learning classifiers this time - SVM, Logistic, Naive Bayes and Bagging classifiers.

2 BACKGROUND

The task of text classification could be done using key concepts in data science like Entity Linking, Knowledge graphs, Coreference resolution and OPEN IE using ReverB. This could potentially be helpful in extracting different features for the classification task. We hoped that it could be helpful for our prediction task. The goal of this submission was to increase the F1 score by making use of meaningful features like Sentiments of the words in the wiki and make sure of several ML classifiers like SVM, Logistic Regression, Random Forest, Decision Tree, Bagging models.

3 APPROACH

Different features are being extracted using different techniques. For this submission, we propose the idea behind REVERB, Coreference, Sentiment Analysis for determining the features. Once we have the feature file ready that is the Final CSV file, we are predicting the death of the character. Along with this we perform some extensive evaluation on the accuracy of some features(Gender) generated by different methods.

(1) **Entity Linker**
As we started with the entity linker for the previous submission, we would like to resolve the problem of disambiguation this time as it was not resolved earlier.
For disambiguation, we make use of BM25 model. Given a query as a character name, the entity linker would make use of BM25 to get all the relevant entities, out of which, the first entity with the highest score would be mapped to the query. For example, Jon can have multiple entities as relevant, but Jon Snow would be the first page that would be coming up with the highest score. So the character name would be mapped to that entity. In a case where the character name does not give any entity during the BM25 search, the search query is changed to the anchor texts for that particular character and hence an entity is mapped.

(2) **Feature Killer (REVERB Implementation)**
In this feature we find whether the character has killed any other character or not. We do this by relation extraction and Argument Extraction. In our implementation we consider the syntactical and lexical constraints. We consider the set of static lexicons such as stabbed, killed, beheaded, strangled, ripped etc. In the end, we come up with a dictionary which has the name of the characters and we mark 1 if the character has killed any other character in the book else 0 (meaning the character has not killed any other character).

(3) **Feature Gender**

(a) **Coreference resolution**
Here we find the gender with the help of coreference. We will search the exact match and precise construct. In exact match, we search the phrases like relation patterns but in coreference, we will search the current sentence and the previous sentence and on the basis of that we will try to refer the names to the exact gender. In current sentence we perform the search from left to right and in previous sentence we perform the search from right to left. We make use of Trees for BFS (breadth First Search) and DFS (Depth First Search) searching in the sentence. Then we come up with a gender map with 1 as male and 0 as female.

(b) **TF-IDF model**
We wanted to improve the performance of the gender predictor by making use of the TF-IDF.
As we have done entity linking, we create a TF-IDF vector for the most relevant passage for an entity(character name), a variation to entity aspect linking. After which we train the characters using the feature vectors from the TF-IDF model to train on using a ML classifier and test it. Despite being a traditional model, we feel this would work better than coreference. We have performed an extensive evaluation on the different Gender predictors we have so far.(including the ones we created during submission 2), which you can find in the next sections.
An example for how this would work is that, if a character name Sansa Stark is passed, the entity linker will give the most relevant page using BM25. From the initial Lucene indexing of cbor, we get the different paragraphs in that particular page. Each paragraph would be considered as a document vector by keeping the query vector as the same(Sansa Stark). The best TF-IDF score is returned and that is used as a feature on Logistic Regression classifier.

(4) **Coreference Nobility**

[*]All three authors contributed equally to this paper.

This is a similar approach to the way we have predicted Gender using Coreference. We find the nobility with the help of coreference. We look for the the exact match and precise construct. In exact match we search the phrases like relation patterns but in coreference, we look for the current sentence and the previous sentence and on the basis of that we try to refer the names to see whether the character is noble or not. In current sentence we perform the search from left to right and in previous sentence we perform the search from right to left. We make use of Trees for BFS (breadth First Search) and DFS (Depth First Search) searching in the sentence.Then we come up with a noble map with 1 as noble and 0 as not-noble.

(5) **Feature Weapon**
The idea behind this feature is to come up with a dictionary with the name of the characters and if they have the most frequently used weapon with them. We consider the name of the characters from Deaths Train and Deaths Test. We train our algorithm on Deaths Train and then apply it to the characters on the Death Test. We do this by searching the weapons in the 5 books and calculate the Frequency of each weapon. Then we calculate the mean frequency. On the basis of mean Frequency we prepare 2 lists. One is the most used weapon and other is least used weapon. Then we do a search for character with most used weapon and least used weapon. Then we will have 2 more lists which are character with most used weapon and least used weapon. Then we calculate Term Frequency of characters with most used and least used weapon. Then we do a cosine similarity with the mean of most used and least used and classify on that basis. We will also be considering to use Entity Linker to disambiguate between the weapon names and the characters to which they are assigned and then do an evaluation and consider the feature which ever performs better.

(6) **Feature Married (REVERB Implementation)**
For extracting this feature we do the same relation extraction and argument extraction. We consider various terms such as weds, married , spouse etc. to find the relationship between the characters. We come up with a list of characters and mark the character as 1 if the character is married else as 0. This helps us in coming up with a new feature which we will use in the final prediction of the character deaths.

(7) **Feature Longevity**
The main idea behind this feature is to see if a character survives more as compared to the other character that is how long will the character survive. So we name this feature as Longevity. For determining this feature we assume that if the character is present in more than 3 books or the character is present in the Book 5 we divide the character into 2 lists short lived and long lived. Then we apply the term frequency on character name. Then eventually we do a comparison with the mean of long lived and short lived and classify long lived characters as 1 and short lived characters as 0.

(8) **Feature Relation Battle Winner (REVERB Implementation)**
In this feature we would be implementing the Battle Winner on the basis of the house and Battles. In this we search for the name of the battles in all the books. Calculate the frequency of the battles. Rank the battles on the basis of the term frequency. Calculate the list of the winning house for the battle listed using the phrases like win/won/defeated. Calculate the list of winning characters using winning phrases like win/won.To increase the accuracy we calculate the winning character associated with the winning house. The character which wins is marked as 1 else it is marked as 0.

(9) **Feature Extraction using Knowledge Graph**
We have improved on our method for knowledge graph that we initially proposed for submission 2. One of the major issues that we have resolved for this submission is character name disambiguation. Currently, our method is able to extract features for only 286 characters out of 917. We observed that disambiguation is the reason for this. We worked on resolving this issue and extract features for as many characters as possible. The approach is to build the knowledge graph on top of our entity linker. We will use the candidate set of entities for each character from our entity linker and map the corresponding entity mention in the text to the most likely character for that mention. For eg: consider two characters "Alyn Ambrose" and "Alyn Estermont", if our entity linker maps the entity mention "Alyn" to "Alyn Estermont", the triple in the knowledge graph containing "Alyn" as subject or object will link to "Alyn Estermont".

(10) **Feature Extraction using Sentiment analysis** We implement a method that performs sentiment analysis and outputs an average positive sentiment and an average negative sentiment for each character. The approach is to make a collection of all the sentences from the text that contains the character name. We further divide this collection into two parts: training and test collection, and train a Naïve Bayes classifier using a bag-of-words model(bag of deadly words eg: cuts, slit, killed, slayed etc.) on the training collection. Further, using the Naïve Bayes classifier, we predict a positive sentiment and a negative sentiment for each sentence in the test collection. Finally, we calculate an average positive sentiment and an average negative sentiment across all the sentences in the collection for every character. We plan to use these two features independently, together and in combination of other generated features and compare the results to know which combination performs the best.

(11) **Extra Baseline methods using mentions**
Additionally, we also use mention count, mention length and mention frequency as features in combination with a different range of other features we generate.
Mention Count: Number of sentences that contain the character name.

Features	F1	SD	ML Classifier
Submission1 Features	0.39	±0.022	Naive Bayes
Featureset2 TFIDF features	0.399	±0.036	SVM
FeatureSet3 Entity Linker features	0.45	±0.025	SVM
FeatureSet4	0.46	±0.022	SVM
FeatureSet1 All features	0.5012	±0.021	SVM

Table 1: Evaluation results with F1 score

Mention Length: Total number of sentences between the first appearance of a character name and the last appearance of a character name.
Mention Frequency: Mention Count / Mention Length.

4 IMPLEMENTATION (EXPERIMENT)

We used both text books as well as the cbor file for extracting features. To achieve our goal we integrated different features from different approaches like sentiment analysis, Feature Gender, Feature Weapon and many more. We integrate various features to get the best F-1 score using Trec Eval. After we generate the features we make a run file. We generate the run files using different Algorithms like decision tree, Random forest, SVM etc. In order to get the F-1 score we make subsets of the run file using the random data sets and calculate the F-1 score for each subset and then come up with mean and standard deviation. We make feature combinations to maximize our F-1 score and then whichever feature set gives us the best result we use the run file for that feature set to calculate the F-1 Score.

5 EVALUATION

- **Feature Variations**

  The graph from Figure 2 shows the different feature variations that we have tried with the above mentioned approach. We started the project with 3 to 4 features in Submission 1 and we developed around 13 features for this Submission.
  - Feature Set1 : Mention Count,Mention Length, Mention Frequency,Avg. Positive Sentiment,Avg. Negative Sentiment,Gender,Weapon,Longevity,Killer Married,Nobility,Battle Winner.
  - Feature Set2 : Gender using TFIDF,Popularity
  - Feature Set3 : Gender,Weapon,Longevity,Killer Married,Nobility ,Battle Winner.
    From Submission2 :
  - Feature Set4 : Gender using anchor texts,Popularity
  - Feature Set5 : Gender using term frequency, Nobility , Popularity , Winner , mother , father

  From Table 1, We can infer that using Sentiment Analysis and the mention count(FeatureSet1) has boosted the score from 0.46 which we got during Submission2 to 0.50 during this Submission. While using entity linker did not improve the score much which can be seen from the drop in the F1 to 0.45 while using the Feature Set2.

Gender Features	F1 Score
Coreference	0.633
Anchor texts	0.55
TFIDF	0.822
Term Frequency	0.62

Table 2: Gender Classifier Evaluation Results

- **Gender Classifier Evaluation**

  In the GOT dataset, the Gender of a person has been one of the key features that was helpful to predict the death of a character. In the previous submission we had two gender predictors and we evaluated them as well.
  For this submission, we compare two different gender classifiers that we have done along with the two classifiers we came up with during Submission2. One of the gender classifiers uses coreference resolution to compute gender, using the dataset as text books while another gender classifier uses the wiki dataset and trains the classifier based on TF-IDF vector and then pass it to a Logistic Regression classifier for predicting the gender.
  To evaluate the gender classifier, the list of training characters are split into two, the first set of 230 characters is passed to the classifier with the actual gender given in the characterdeaths.csv, and the prediction is made for the rest of the 231 characters.
  The graph in Figure 2 infers that using a TF-IDF model works better than using the customised coreference system, with an accuracy of 0.70 and a F1 score of 0.83.

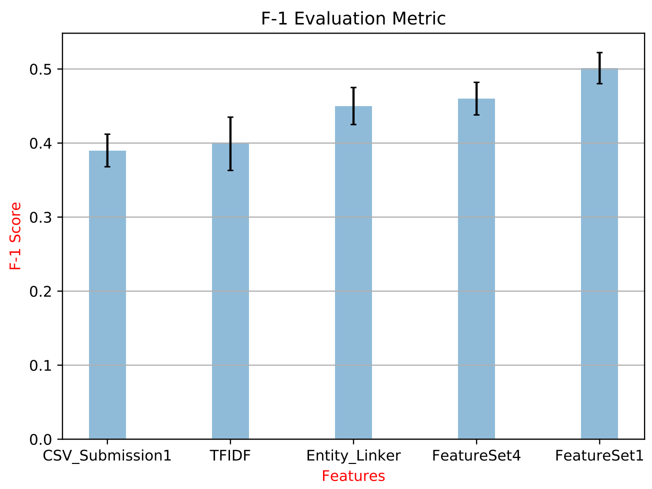

Figure 1: F-1 Score Evaluation for Death Prediction

6 WORK DISTRIBUTION

The work distribution is as follows: -

- Abhinav Gupta

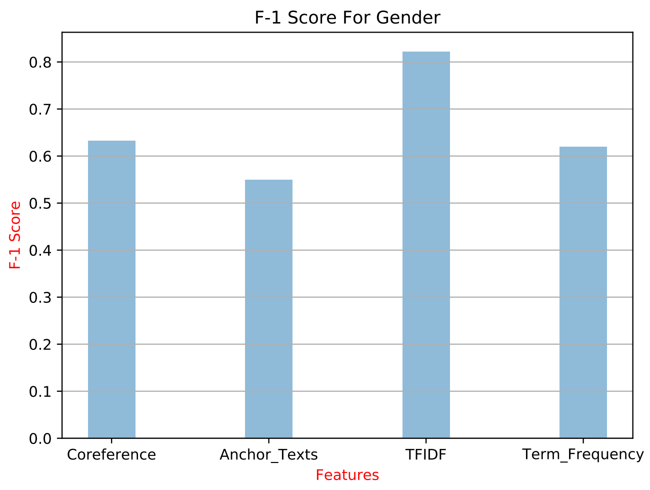

Figure 2: F-1 Score Evaluation for Gender

  - Feature extraction methods.
  - Making use of REVERB and Coreference functionality
  - Running feature set on ML methods like Bagging classifier and Ridge Classifier.
  - Making subsets of different features and evaluating which feature set gives the best F-1 Scores.

- Stavan Anjaria
  - Knowledge graphs
  - Sentiment Analysis based features.
  - Baseline features.
  - Integration.

- Gayathri Venkatasrinivasan
  - Improve the entity linker by resolving disambiguation.
  - Improve the Gender predictor.
  - Worked on different Machine Learning models.
  - Performed the extensive feature analysis to see which features work the best, based on that feed those to the ML classifier to get the best F1.
  - Articulated the report.

7 SYNERGIC EFFECT OF THE TEAM

As a team we were be able to achieve a better F1 score by incorporating Abhinav's features, some of the features like Gender using TFIDF, Gender using the anchor texts of entities, Popularity using inlinks are developed by making using of Gayathri's entity linker along with Stavan's features for Knowledge Graph and Sentiment Analysis.

8 CONCLUSION

Our features have performed better than the features we had used during Submission2. Out of the all the features used, Sentiment analysis and TF-IDF based features stands out with improved results. A future work would be to dwelve more into some of the Open IE methods and Relation Extraction methods.

REFERENCES

[1] Shen, Wei, Jianyong Wang, and Jiawei Han. "Entity linking with a knowledge base: Issues, techniques, and solutions." IEEE Transactions on Knowledge and Data Engineering 27, no. 2 (2015): 443-460.

[2] Nickel, Maximilian, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A review of relational machine learning for knowledge graphs." Proceedings of the IEEE 104, no. 1 (2016): 11-33.

[3] Fader, Anthony, Stephen Soderland, and Oren Etzioni. "Identifying relations for open information extraction." Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 2011.

[4] Federico Nanni, Laura Dietz Entity-Aspect Linking: Providing Fine-Grained Semantics of Entities in Context