Submission 3

Team 7
Sai Arvind Reddy Desireddy
Medhini Shankar Narayan
Akhila Bezawada

1. Introduction

Our shared task is to find which characters died in the Game of Thrones series. This is an interesting Data Science project as we could analyze and establish the facts about each character and make prediction about who dies/survives, based on pattern and knowledge acquired which involves hundreds of characters which are introduced and evicted through 5 published books. We are Developing a system or model which is capable of predicting accurately whether a character dies/survives.

We divide this task into many subtasks which involve generating features to our models and trying different Data Science methods that are learnt in this class to generate our features. We have focused on Entity Linking/Disambiguation, Coreference and Open IE. Through these methods we have generated six features: Gender Predictor, Gender Predictor using Tf-idf, Two Allegiance predictor, Relation Count and Popularity.

Our goal being to predict character death, these features are kept into a prediction model. We take those models from 'sklearn'(Machine Learning models). In this submission, we are implementing custom Machine Learning models that would take our features as input and will give us an F-1 score. We are also trying to improve our F-1 score compared to previous submission.

2. Related work

2.1 Disambiguation

Goal of Entity linking/disambiguation is to tie together named entities to an identified object or concept. From the video link of Professor Laura Dietz, https://www.youtube.com/watch?v=CBvE3BNErgE we have gathered our method of approach from that. Over the course, we have studied papers [Hoffart et al. Shen, Wang, and Han, Moro, Raganato, and Navigli] that help us in understanding of the concept. For example: 'David' is the governor of the state. Entity linker should map 'Dave', 'Davis' (other names of David) to 'David' as itself who is the governor of the state. For Entity linking to work, knowledge base is used.

[Shen, Wei, Jianyong Wang, and Jiawei Han] in their paper, they divide their task into entity generation and ranking. Candidate entities are identified and generated. They consider their knowledge base to be Wikipedia. After candidate entities are generated, they are ranked using supervised and unsupervised ranking methods.

2.2 Open IE

Open IE refers to the extraction of relation tuples from plain text. Relation names is just the text linking two arguments. Through the case, we have read papers which describe the approach towards Open IE. [Fader, Anthony, Stephen Soderland, and Oren Etzioni], in this paper assertions are extracted from massive corpuses. Reverb (Open IE) is compared with Text Runner. Approach focuses on phrase patterns in terms of POS tags. In our approach of Open IE we are using Stanford Parser, where each sentence is split into set of entailed clauses. Each clause is shortened producing shorter sentence fragments. These fragments are then segmented into Open IE triples and is then given as output by the system.

2.3 Coreference Resolution

Coreference resolution is the task of finding all expressions that refer to the same enitiy in a text such as “Mary” and “she” refer to the same person. [Raghunathan, Karthik, et al], in this paper a sieve framework is used where each model is built on previous model’s cluster output. Documents are parsed using Stanford Parser and a series of independent coreference models are used. Each model has different rules to link two mentions. This method is used in our allegiance predictor and gender predictor which are our features for the model.

2.4 Tf-idf

Term frequency-Inverse Document frequency is a numerical statistic that reflects how important a word is in our document. As we learnt in Information Retrieval course, tf-idf value increases proportionally to the number of times a word appears in document. Using tf-idf we generate one of our main features- the gender predictor. For every paragraph from the Wiki page, tf-idf vectors are constructed, more details about the gender predictor approach is explained later in the document. Even for disambiguation, we use tf-idf methods.

3 Approaches

To generate features for our model, we use the below approaches. Depending on our needs we have tokenized and removed stop words in cases and some we haven't. In some parsers, we have not removed stop words based on our usage.

3.1 Disambiguation/Entity Linker

We use Game of Thrones cbor file for this approach. Parsing the cbor file, a mapping of all textual references in the Wiki for all individual pages is created. We identify the pages that can act as entity definition (Stanford parser will identify entities in a sentence. Those are cross checked with Cbor names) where 'he' or 'she' with every character name is combined and given to lucene. Tf-idf methods are used to disambiguate. Whole para text (sentence) plus character name is given as input and we find the correct para related to that based on tf-idf score for that reference.
Disambiguation is handled by using BM25 as well, where the page name is having multiple pages referred to it, which we try to get it resolved. So BM25 is used to get the relevant page names for an entity and it maps the first ranked page to this entity. If we are unable to identify identities using Stanford, then we do the following method.
We identify all entity mentions in text and for each mention we retrieve candidates. Entity is identified with surrounding text and search query now will have all those that was identified. We are using tf-idf methods to disambiguate. Example: Sansa is a disambiguation name for Sansa stark entity. Sansa will have inlink from sansa stark, and anchor text.

3.2 Open IE

From the paper which we studied in class; we came across The Stanford Parser. Incoherent extraction is when extracted relational phrase has no meaningful interpretation. Syntactic constraint eliminates incoherent extractions. It does a simple parts of speech tagging where it limits the relation to be either a verb (e.g.: invented) , a verb followed immediately by a preposition (e.g.: located in), or a verb followed by a nouns, adverbs or adjectives ending in a pronoun(e.g.: has atomic weight of). If there are multiple possible matches in a sentence for a single verb, the longest possible match is chosen.
Finally, if the pattern matches multiple adjacent sequences, we merge them into a single relation phrase (e.g., wants to extend). This reduces number of incoherent extractions and covers relations expressed via light verb constructions.
We obtain Relation Count as a feature from this method.

3.3 Coreference

We are using the Stanford parser to implement this as a method. Each paragraphs/ sentences are parsed accordingly and coreference names are replaced with original entity names. We find all the expressions that refer to the same entity. This method is then used in our gender predictor and allegiance predictor for parsing.
Using co-reference, we would be trying how it works on gender. We are not sure if this will come up to our expectations but still giving out our best try. We could achieve an accuracy of 66% and with the help of coreference the score may have a bit of impact.

4 Features

We have 6 Features: Two Gender Predictors, Two Allegiance Predictor, Relation Count and Popularity. We believe that these features contribute in increasing our F-1 score. Gender, relations, allegiance, popularity for a character will add the chances for their death. Suppose the character is male, belongs to a famous house, has many relatives and is a popular character: he/ she will die. That is the myth. We have our methods to checks if the myths are true or not.

4.1 Gender Predictor 1 (GP1)

We use Wiki pages and cbor files for this approach. For example: if we need to predict "Robert Baratheon" is Male/female, then we go to the GoT wiki page of Robert Baratheon title of the page and redirect anchor text, if this character name is matched with any name present in the Cbor file then it would be the wiki page of Robert Baratheon.
We take the page and first paragraph of that page and build a tf-idf vector, we do the same for all the paragraphs in the page and build a tf-idf vectors based on the words in that paragraph. We will do this for the training characters. We take these names and match them thoroughly with anchor text, redirect and titles. We repeat the same with test data i.e; We will take all testing characters. We take these names and match them thoroughly with anchor text, redirect and titles.
Then, we will train tf-idf vectors from the first paragraph and then train it by using logistic regression since it is a classification model which is used, in our case, to predict the binary output deciding whether the character is male / female.
Once we get all the tf-idf vectors for male and female we can see which one has more features, whether it is male/female, we also have Robert Baratheon tf-idf vector, we can predict whether it is male/female.
Output will be:

Character name	Tf-idf score	Male/ female

4.2 Gender predictor 2 (GP2)

As stated above, we have chosen gender as a feature because we believe the fact that death rate of male is more when compared to female. So, we are interested in doing gender prediction using query based.
We parse whole 5 textbooks using lucene using cbor and our query will be Character name + male pronouns, Character name + female pronouns. We will take the top scoring results from the input queries and based on score we can decide if the character is male or female. For example: ‘Arya stark + he’, ‘Arya Stark + she’ will be our query, if we get top score for ‘Arya Stark + she’ we decide that Arya Stark is female.

4.3 Allegiance Predictor 1 (AP1)

We consider every sentence in cbor. For every sentence we find entities (character column will be our entities). We eliminate certain scenarios. Suppose an entity is followed by an allegiance name, most of the cases are eliminated. Next case we consider is an entity and allegiance present are in same sentence. 70% of the cases get eliminated here.
Our next try was nearest allegiance, name near to the entity. Suppose for example, let’s take 2 sentences into consideration. “x is in A house and B is also a house. A house has x and C also is there.” We search through fandom and we see that entity x is followed with “A” twice in the same sentence and hence we assign entity x with allegiance A. We needed to map the bigger house names to its smaller house names, so we used Wiki ice and fire for mapping with cbor. We did web scraping to do that.
We do this with tf-idf. We also need disambiguation for this, we need to link Stansa with Stansa stark. We need to remove the disambiguation name and replace it with original name.

4.4 Allegiance Predictor 2 (AP2)

We tried the next allegiance predictor using Bag of words model. It uses a vector space model, we can represent unstructured text where each dimension of vector is a specific vector/ attribute.
Bag of words model represents each text document as a numeric vector where each dimension is a specific word from corpus where value indicated frequency in the document.
Each column will represent words from corpus and each row represents documents. We find the count of specific allegiance names using this model and assign it to the entity

4.5 Popularity (PO)

This is used as a feature for our prediction to generate the popularity of a character i.e., whether the character is popular or not popular. Entity Linker is used to create inlinks and outlinks of the graph from cbor file i.e., a character name. We find the popularity score for each character where we implement page rank algorithm.
Page rank is used generally to rank web pages based on search engine results. We also find the popularity score for the same. From page rank algorithm we take inlinks, outlinks that are to be considered from our cbor file.
We reimplemented this feature again to make this a main feature for consideration. We give a popularity score to each character that will be used as a feature.

4.6 Relation Count (RC)

In each sentence, we remove pronouns and we bring it to the subject predicate object stage using Open IE from Stanford Parser. We then get relations for every entity and we put a count to it (only close relations are considered like father, brother, sister…). For example, x has a mother, father and son. So, x's relation count is 3. Through this feature we can know if more the relation count, his chances of survival are less or more.

These are the six features that we have obtained from our methods. Next, we train these features on Machine Learning models. This submission we have tried some custom ML methods.

5 Custom Machine Learning methods.

We implemented 3 custom ML methods i.e; Linear regression, K-means and Null regression methods from beginning. We did not use the inbuilt functions.
→Linear regression defines relationship between dependent variable (Y) and independent variable (X). It can be written as y= mx+c, where m is the scalar and c is the bias coefficient. We can find these coefficients using Ordinary Least Mean Square method and Gradient Descent method. We calculated mean, covariance and variance of the data. Using this we have calculated 'm' and 'c'. This will give us score. We have done this for name, GP1 and name, GP2.

m = sum((x(i) - mean(x)) * (y(i) - mean(y))) / sum ((x(i) - mean(x))^2)
c = mean(y) - B1 * mean(x)
where mean(x) = sum(x) / count(x)

→ K- means: we have used this method to predict our dead or alive problem. We create a K- means transformer which can be used like a built in sckit learn transformer.

→Null Regressor: we do random guessing in this method where mull model would be to take mean of training data and using that for every prediction. This model tells how well our current model is working. If most of the data is zero, then it predicts zeros and if most of them are ones it predicts ones.

From other modeling capabilities of sklearn, other classification models that we used in this submission are Decision Tree, Random Forest, Logistic Regression, XGB classifier Each model was trained with various features. According to us, more the number of features higher is the F-1 score. Training with more features might also lead us to overfitting which might have an impact on the test set. To determine which feature matter and which don't we do ablation test on the models with all features that is discussed in further sections.

6 Results and Evaluation

From all the features we generate a CSV file, where its six columns are the six generated features. We call this feature as our 'feature file'. Our new features generated in this submission are GP2, AP1, AP2, RC. Other two features (GP1, Po) we have tried improving from the first time.

6.1 Evaluation of Gender Predictor 1 (GP1)

Evaluating our Gender Predictor which was done using tf-idf. Our Gender output had character name, tf-idf score, Male/Female (0,1). We divided this output into test and train data and modeled using the Decision Tree classifier. Our output had 1/0 (Male/Female). This result was then fed into our feature file.

Feature	**Classifier**	**F-1 score**
Gender Predictor 1 (GP1)	Decision Trees	0.821

Table 1

The above table (Table 1) indicates that our Gender predictor has a F-1 score of 0.821 when used with Decision tree. We had expected a high value as this method takes tf-idf values and it's a popular method discussed in class. We learn that our Gender predictor using tf-idf does a good job in predicting male/female.

6.2 Evaluation of models

Keeping all the features, we try various Machine Learning models along with our custom methods. Table 2 shows various models with the F-1 score considering all the features.

Model	**F1 +_ std error**
Decision tree	**0.542 +- 0.03**
Random Forest	0.478 +- 0.04
KNN	0.416 +- 0.05
Bagging classifier	0.434 +- 0.04
XGb Classifier	**0.543 +- 0.03**
Custom 1 (Linear)	0.496 +- 0.03
Custom2 (KNN)	0.486 +- 0.04
Custom 3 (Null)	0.497 +- 0.03

Table 2

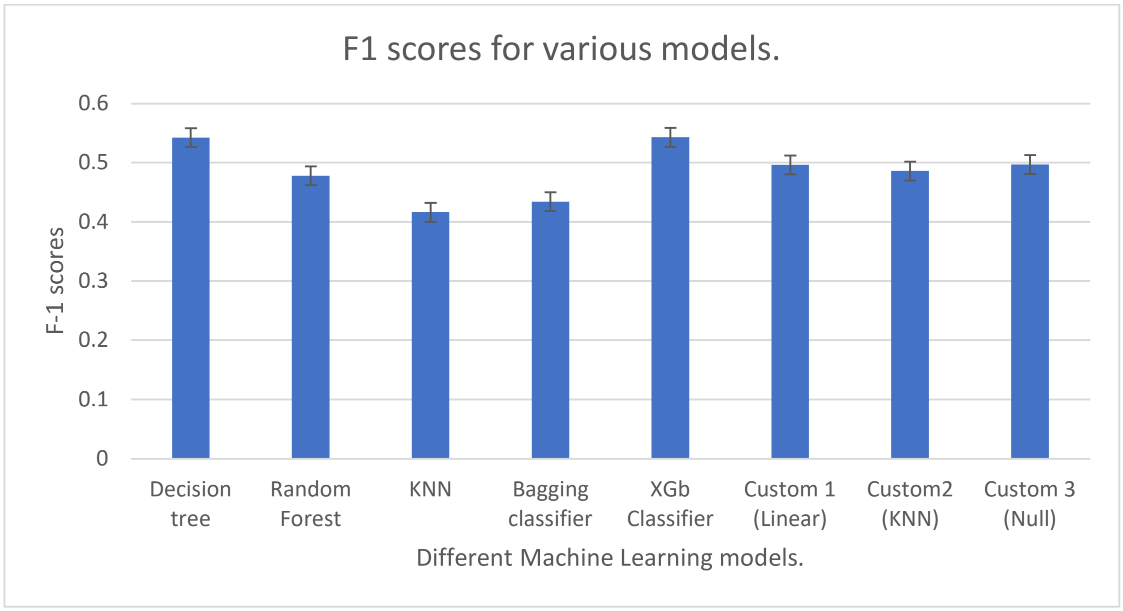

Fig -1

The above graph (Fig- 1) shows various F-1 scores for Machine Learning models and we can see that highest F-1 of 0.542 was obtained for Decision Tree. This was considered with all six features. We expected Logistic Regression to do better, but instead Decision Trees and XGB classifier gave us better results. We can also see that we tried two variations of KNN and custom KNN is giving better results than the normal one.

6.3 Ablation Test

We have six features; we need to find which features are actually useful and which will lead to overfitting. We remove a feature from the dataset, we find f-1 measure. By removing a feature, if our f-1 goes down we can conclude that those features are important. Instead, if F-1 rises up after removing features, we can say that those features do not add contribute as important features. We find that if we remove features AP1 and GP1 separately, f-1 scores goes down. This shows that they are an important feature. If we remove PO feature, f-1 score is not affected. We learn that this feature plays insignificant among other features. If we remove GP2 and AP2 our f-1 scores shoots up. This means these feature acts as dummy features and may lead to overfitting. Since we got Decision tree as our best method from the above results, we did ablation test with Decision tree model.

Table 3 depicts the results of ablation test. 'All' means when all features are included. 'All - GP1' means when GP1 feature was removed from entire dataset. This was carried out with Decision tree model with varying features.

Features	**F1 score**
All	0.542
All - GP1	0.451
All - GP2	0.672
All – AP1	0.471
All – AP2	0.615
All - PO	0.532
All - RC	0.478

Table 3

7. Discussion

In this submission, we generate more features to see how well the model performs. We believe some features lead to overfitting, so we don't get desired values. For some methods, we get good results whereas some methods did not do well at all like Logistic Regression.

As a team we could help in improving F-1 score by generating more features and examining how they work. We could also try different methods that we studied during the course over this project like entity linking, coreference resolution and Open IE.
In future we would want to do sentiment analysis, knowledge graphs to extract more features and examine how it would impact on f-1 score. This has been a challenging and exciting project for us to work on

8. 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. Raghunathan, Karthik, et al. “A multi-pass sieve for coreference resolution.” Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2010.

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. Entity Linking video: https://www.youtube.com/watch?v=CBvE3BNErgE