Predicting Death In Game of Thrones

Soheil Gharatappeh

April 22, 2020

1 Introduction

These days Machine Learning tools are being widely used to predict different phenomenas. From the stock market, to autonomous cars, researchers are interested to have their prediction on all aspects of life. Recently, the movie Game of Thrones has gained a huge popularity, and everyone was wondering *which of the characters can make it to the end?*. This could be a great subject for a research project in Data Science realm. There are various resources of information, from structured wikis to un-structured text such as the books of this legend. We can base our prediction on information that is coming from all of these resources. However, using the right information helps to put all pieces of this puzzle together.

One rich and valuable piece of information we have access to is a data set obtained by the fan's community. This data set contains various information, from the date of birth, to the allegiances, nobility, gender, etc., and can be used to train different machine learning methods, in order to find out what aspects of the information is the most influential in the chance of death for each character.

For the next phase in the project, we would like to omit the data set, and work only with the wiki and un-structured text of the books. This means the project will shift from a purely machine learning project, towards a more realistic practical data science one, where we need to obtain the information we need, clean it, filter it, and then use that information as a single feature for the machine learning algorithm. Obtaining information from un-structured text is very hard, due to the noisy and nonlinear nature of natural language.

In order to tackle this issue, we can use knowledge bases. A knowledge base is a big set of triple entities (two names connected with a relation), obtained from a structured text such as wiki, and contains rich information about the world's entities, their meanings, and relationships. Knowledge bases give structure to the raw and noisy data obtained from the web, and makes them usable for a machine. A crucial part of this process is entity linking, where a named entity is linked to the knowledge base.

This work is heavily based on the idea of entity linking and knowledge graphs. In the following sections, we first introduce entity linking and review a few related works in this area. Then, we explain our approach to the problem of using entity linking in order to predict the chance of death for a GoT character, and at the end, we evaluate our work using different measures.

2 Background

Knowledge sharing platforms such as Wikipedia, play an important role in creating machine-readable knowledge bases. Knowledge bases contain rich information about the world's entity, and their relationships. Using KB's, one can extract useful information from the raw and noisy data in a text. Entity linking is a key element of this process, where a named entity is linked to the knowledge base.

The entity linking is a challenging task. Name variations, aliases, abbreviations, alternate spellings result in entity ambiguity. The task is to find a map from between a set of pre-identified textual entity mentions m into their corresponding entity in the knowledge base. There, also, might be some entities in the text with no entity record in the knowledge base (unlinkables or NILs).

The process starts with name entity recognition, where the boundaries of the entity is identified. There are three steps in the task of entity linking; candidate entity generation, candidate entity ranking, and unlinkable mention prediction. In the first step, a relevant subset of entities are retrieved from the knowledge base. Then, we need to rank them in the next step, so that we know which node in the knowledge base is more relevant to our entity and which one is less relevant. And, at the end, we need to have a strategy to deal with unlinkables [4].

In order to evaluate each of these algorithms, a few metrics have been introduced in [4], such as precision, recall, accuracy, and F1-score. Also, some data sets are published in [4] in order to be used as the ground truth for this field. Going through the list of all available data sets is out of the scope of this writing.

In another related work, Cheng et at. introduced Relational Inference for Wikification [1], where they are trying to disambiguate entities in a text by identifying the context of it, and relating an entity to the most relevant Wikipedia page. The idea is to use the relational structure of the text in order to generate more relevant title candidates, rank them better and deal with unlinkables more efficiently. They form an Integer Linear Program using the document and the knowledge base consist of triples, and solve it to find the best assignment for each title-surface set (where surface is an entity mention). This approach is evaluated in comparison to other Wikification systems such as Milne and Witten (M&W), and Ratinov et al. (R&R) by using data sets from GLOW system, a work by Ratinov et al. and evaluating it with Bag of Titles (BOT) F1 measure. The results suggest a good level of improvement over all 4 different data sets, in comparison with M&W and R&R. They also compared their approach to top 3 TAC 2011 systems, where they used B^3 and Micro-Average as the evaluation metric. Their system performed comparatively similar to the top 3.

Also, in [2], Ferragina et al. talk about adding Wikipedia links to the respective mentions of entities in a text. The contribution of this method is its applicability to short or poorly composed texts, such as tweets, snippets of search engines, etc. The idea is to use Wikipedia anchor texts as spots and pages that are linked to them as a possible meaning for each entity. The ambiguity is solved by using a fast and effective scoring function, which exploits the sparseness of the anchors in a short text. One important phase in disambiguation is prunning un-meaningful anchors by using the link probability and coherence of a candidate annotation. This approach is evaluated in comparison Milne & Witten method on three different data sets, the first one is derived manually, and the second and the third is from Wikipedia. They evaluating their method with F measure, precision and recall. The results show that tagme outperformed in short texts, and have a good comparative performance on long texts in compared with Milne & Witten method.

In [3], Mendes et at. explain their approach to automating the process of text document annotation with DBpedia URIs. The goal is to find and disambiguate DBpedia mentions automatically and adaptively, for the needs of each user, specifically. The idea is, to recognize the entity in a sentence, first. Then,to map the entity to a selection of candidates from DBpedia. Next, using the context around the entity, the disambiguation stage is proceed. And, at the end, we let the user to customize the resulted annotation for his need through the configuration parameters. They evaluated their disambiguation strategy on some unseen DBpedia resource mentions from Wikipedia. They used accuracy for evaluating their system, and it suggests a good upgrade in compared to all baseline methods. But, unfortunately, they didn't provide any accuracy results for other publicly available approaches, and they only compared their algorithm with those methods in terms of precision-recall paradigm. They also tested their system on a set of manually annotated news article, and compared their annotation with other publicly available annotation systems. They evaluate their algorithm using F1 measure and compared it to other methods. But, as the result suggested, they could not outperform the wiki machine. But, they did better than the others in terms of F1 score.

3 Approaches

3.1 Entity Linking

In this section, the approaches for tackling the problem of predicting the death of a GoT character is explained. One of the main ideas were to first train a perfect classifier using the data set, and then discard the features that are read from the prepared data set, and extract those features purely from the books. Although the idea seemed appealing and sound at the beginning, it had a big flaw, and that was; extracting features from an unstructured text is a drastically hard task, if not even impossible. This is where having a knowledge base, and doing entity linking over it can be helpful. Using a knowledge base (that is obtained, for example, from the GoT's Wiki), we can employ entity linking in order to disambiguate entities. Entity linking can be helpful in constructing a big feature vector for each entity (each attribute in the Wiki can be served as a feature), which can later on be used to train a classifier. The more accurate entity linker we use, the better result we get from the classifier.

However, having a complete knowledge base is essential in the process of entity linking. It is usually the case that because of having multiple entity in the knowledge base corresponding to the surface, we need to rank the entities so that we can retrieve the most relevant entity to the user. But, unfortunately, that is not the case in here. There is a limited number of wiki pages for each character, with a very limited information. Only 263 of characters of the data set have a corresponding node in the knowledge base. Therefore, I did not include the entity linker features into the set of features.

3.2 Gender Prediction

Due to the natural ability of English Language in specifying the gender of a noun phrase in a sentence, there seems to be a possibility for predicting the gender of a character. This gender predictor is based on the use of pronouns in the sentences surrounding a character name. Two methods are employed for determining the gender.

First, we use a pronoun counter to make the decision about the gender of character. We count the number of pronouns in the sentence that a name has occurred, plus the sentence that comes immediately after. The gender is determined by the dominant pronoun.

Second, we apply closeness measure to predict the gender. The gender of the closest pronoun is assigned to each corresponding noun.

At the end, we used the results coming out of this rule-based classifiers as a set of features in order to train a few classifiers. Classification methods that has shown to be promising on this project, such as decision tree and KNN, as well as a logistic regression classifier is selected to be run on this data set.

3.3 Features

It is very important to determine which features in the data set are the most useful ones. Factoring out the less informative feature eliminates noisy data and helps the classifier to gain a better accuracy, reducing the chance of overfitting, to obtain a better generalization from the data. The problem of finding the best features of the data set is heavily tied with our ability to extract accurate features from the knowledge base and the book. For instance, one can come up with a set of features, with a high accuracy. But, if extracting those features from the text and the wiki is not feasible, then we have to eliminate them.

Based on various experiments, we figured some features are more important than the others, and here is a set of features are used to train the classifiers

1. Mention counts
2. Gender
3. Proximity to deadly words
4. Last n mentions
5. Graph of connected characters

Now, we explain the features we used, and what is the intuition behind choosing them and how we think they help in predicting if a character is dead or not.

3.3.1 Mention Counts

The number of times a character has appeared in the book is an important measure that shows how crucial was the character in the story. This feature can imply how a character plays a centric role, and if it does, how centric roles are at the risk of dying. In addition, the number of times that a character appears in each of 5 books can also be important, as it can be an indicator of when a character has stopped being mentioned, which can be translated into the death of a character.

The total number of times that a character has appeared with full name in the text books is used in here. We also have a 5-column feature set of mention counts per book.

3.3.2 Gender

Gender is another contributing factor in the death of a GoT character. It's more likely to be dead when you are a male in this story. Therefore, we use the frequency of the pronoun and the closest pronoun to create two sets of features, called gender feature.

3.3.3 Proximity to Deadly Word

It is common sense that if a character appears around words that imply death is more likely to die. The character is probably a warrior, or very close to the center of tensions. That is why we think occurrence of a character with a deadly word in the same sentence can make a good feature.

Counting the number of occurrences of a character around deadly words is one measure that we employed to extract a feature. However, it is not the most accurate one. Tf-idf is another measure of closeness in the field of information retrieval, which has shown more promising results. To obtain the proximity of a character to deadly words using tf-idf, we first find the sentences that a character appeared in the book, and create a document out of these sentences. Then the tf-idf score of this document is calculated against a sentence containing all of the deadly words (a hand-crafted list of words prepared for this matter which contains words like stab, kill, die, etc.) and the character's name.

3.3.4 Last n Mentions

Inspired by the proximity to deadly words features, we created another set of features consisting the number of times a deadly word appeared in the last n mentions of a character. We simply used the deadly word count to create these features, and tried different ns, and picked 1 and 5 that showed better performances.

3.3.5 Graphs

The relation between characters, being brother, sister, enemy, friend, ally, etc. with other characters is another key factor in the chance of dying in this story. This network representation of characters help us to extract important features from the graph of characters such as betweenness centrality, closeness centrality, degree centrality, Page Rank (which characters are most referred to).

The graph, however, can be created in different ways. Two strategies that we used in this project was as follows:

a. Characters are related if they appear in the same sentence

b. Characters are related if they appear in the same page in the knowledge base

4 Evaluation

In this section, we evaluate our work in terms of F1-score metric. The data set of 918 names is divided into training and test sets. The rows in odd indices are used as the training set and even rows are used for testing. Figure 1 shows the performance of different classifiers on this data set. As shown in this figure, decision tree was able to capture the trends in the data better, and predict more accurately. In addition, by looking at figure 1 one can immediately conclude that linear models didn't work well on this data set at all. This shows there is no linear relation between the features and the response.

4.1 Gender Predictor

In another experiment, the rule-based gender classifiers explained in section Approaches trained and tested using the same data set as the previous experiment. In this experiment, at first a frequency based model is tested. This model could obtain a f1-score of 0.902. Then, using the closest pronoun policy, another gender predictor is run against the data, which obtained f1-score of 0.623. After running these rule-based classifiers, I treated them as a set of features and used them to train some

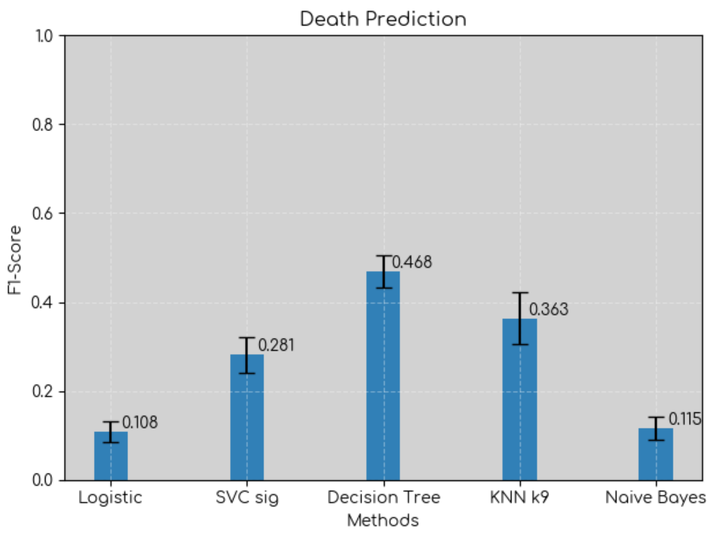

Figure 1: F1-score of various classifiers on a set of features completely obtained from the text.

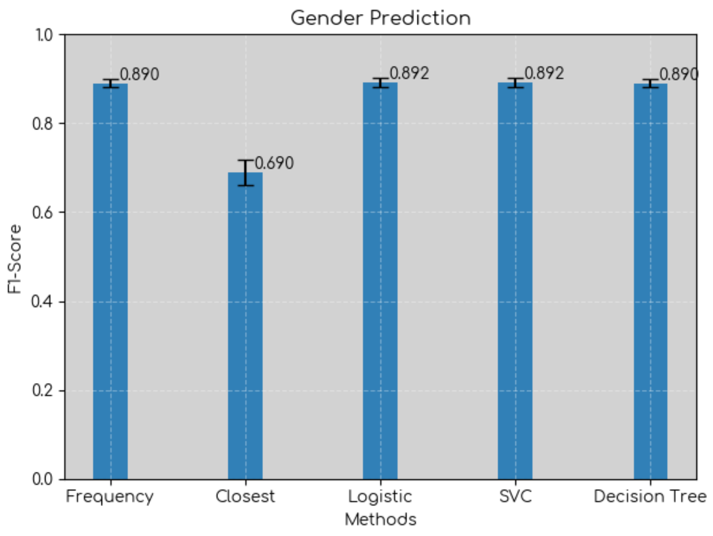

Figure 2: F1-score of gender classifier. The first two bars are related to the rule-base predictor, and bar 3-5 are f1-score of different classifiers using those rule-base feature for training.

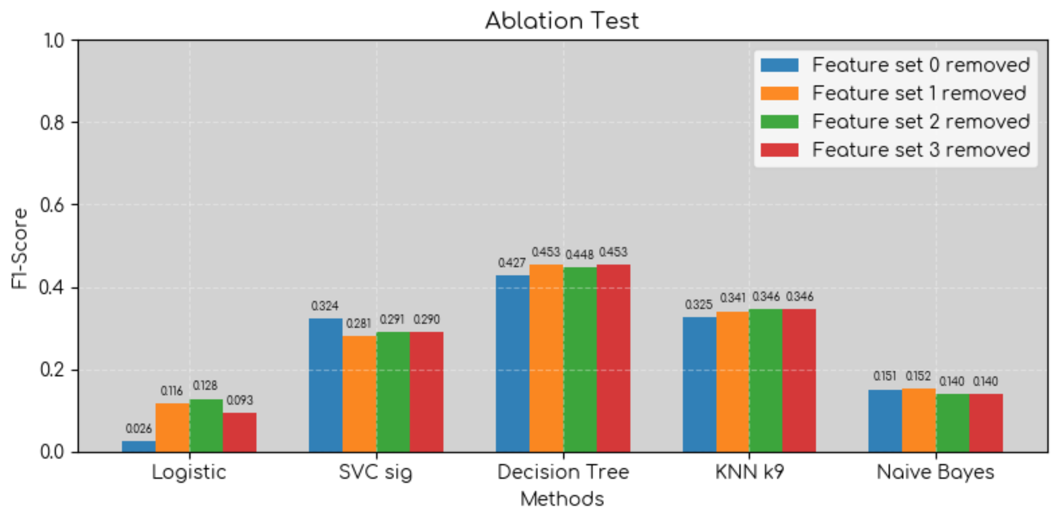

Figure 3: Ablation test across various classifiers.

standard classifiers, such as SVC and decision tree. The result of this experiment is shown in figure 2.

Interestingly, classifiers could not obtain a f1-score higher than the frequency based method. And, also the type of classifier did not make a difference in this case. We can only conclude that the features are highly correlated, and adding the second feature did not improve the performance of the classifier.

4.2 Ablation Test

After adding a lot of features to the training set of a classifier, the next step is to figure out were those features actually useful in the final prediction, or they only add noisy data to the system and cause it to overfit. A great way to show how good and useful was a set of feature is to do ablation test.

In ablation test, we remove a feature from the whole data set and calculate the evaluation metric (f1-score in our case). If f1-score of the newly created data set is more than the original data set, the feature that we just removed was bad and useless. So, we can remove it from the system to increase the accuracy of prediction. On the other hand, if the f1-score of the newly created data set decreases, it shows that we actually removed a useful feature.

One conclusion that can be drawn from figure 3 is that the first set of features, which are mention counts, are highly linearly correlated to the response. When removed from the data set, logistic regression loses its ability to predict accurately, but, SVC actually benefits from not having it. Note that SVC with sigmoid kernel is a nonlinear model, and is not able to capture a linear relation well.

Another conclusion is that decision tree has performed better in general in compared to all other methods.

And, last but not least, there is no evidence that removing a set of feature could increase or decrease the f1-score. So, we keep all of the features in the data set.

4.3 Best Subset Selection

We can extend the idea of ablation test to what is called Backward Best Subset Selection. In backward best subset selection the classifier starts with a full set of features, and in each step a feature is removed and the evaluation measure is computed. Next, the feature corresponding to the highest f1-score is removed from the feature set, since removing that feature leads to an improvement in f1-score. This procedure continues until removing a feature does not lead to an improvement. However, due to the heuristic nature of the algorithm it only gives a set of features that locally maximize the f1-score. This means there might be a better subset of feature set that result in a global f1-score maximization.

In this project, however, due to the randomness of the best-performing algorithm, which is decision tree, we obtain different results across runs. And also, the difference between f1-scores that get removed in each step and the scores that make it to the next step is so low, that makes the comparison between them meaningless. Table ???, however is one of the results that we obtained running Backward Best Subset Selection.

5 Future Work

5.1 Coreference Resolution

Another problem to address in this project is disambiguation. A lot of names in the book are not fully written, and that makes them ambiguous. Or, they may be referred to by a pronoun. The way we currently define an entity mention is if a) the name is fully appeared in a sentence b) one part of the name appears in a sentence. Extracting accurate features such as number of mentions from an unstructured text, such as GoT's books is usually very hard in this situation. This is due to the ambiguous nature of unstructured texts.

In Coreference Resolution the goal is to find different instances in a text that are referring to the same entity name. Employing coreference resolution, the accuracy of features can potentially increase, since our feature are more realistic. However, due to some technical issues in the text books, running coreference resolution was very hard. There are several instances of missed quotation marks, that makes coref resolution fail.

5.2 Relation Extraction

The problem of figuring *If character x dies?* can be represented as an Information Extraction problem. Information extraction algorithms, such as OpenIE extract relations from an unstructured text using Freebase. This relations are in a form of tuples `<entity1, entity2, relation>`, where the relation is selected from the huge predefined set of relations. However, in this project the focus is on finding a very specific relation, a relation that is revolved around death. So, we are mostly interested in relations such as `<character1, character2, isKilledBy>`, or `<character1, died>`. Extracting this relations from an unstructured text is not very hard when you have access to an accurate coreference resolution. The fact is all of this death scenes have been explained in somewhere in the books. But, the ambiguous nature of language makes it very hard to find where this information is hidden. Especially when a name consecutively is referred to by a pronoun.

References

[1] Xiao Cheng and Dan Roth. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[2] Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, page 1625–1628, New York, NY, USA, 2010. Association for Computing Machinery.

[3] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011.

[4] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, Feb 2015.