Predicting Death In Game of Thrones

Soheil Gharatappeh

March 14, 2020

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.

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.

Approaches

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.

Gender Prediction

It is very important to determine which of features in the data set are the most useful ones. Factoring out the less informative feature eliminates noises from adding to the data and helps the classifier to gain a better accuracy. The problem of finding the best features is heavily tied with the problem of which features of the data set can be extracted from the knowledge base and the book with higher certainty. One can come up with a set of features coming out of a regularization method (such as lasso), with a high accuracy. But, if extracting those features from the text and the wiki is not feasible, then we have to eliminate that feature.

Due to the natural ability of English Language in specifiying the gender of the subject in a sentence, there seems to be a possibility for predicting the gender of a character. A gender predictor is implemented using the pronouns to predict the gender of a character in the sentence they are mentioned. Two methods are employed for determining the gender. First, we use a pronoun counter to make the decision. We count the number of pronouns in the sentence that a name occurs and the

sentence after, and the gender is obtained by the dominant pronoun. Second, we apply closeness measure to predict the gender, and we assign the gender of the closest pronoun to each corresponding noun.

Features

Based on various experiments, we figured out that features such as the gender, the mention frequency, mention of the name in different sections of the book and how the mention count distributed among different books, and proximity of the character's name in the book to some deadly words are the most useful features. Therefore, we employ various tools to obtain a prediction from the above-mentioned features. To obtain the proximity of a character to deadly words, we first find the sentences that a character appeared in the book. Then the tf-idf score of this sentences (deadly sentence) is calculated against a sentence containing all of the deadly word (a hand-crafted list of words prepared for this matter which contains words like stab, kill, die, etc.) and the character's name. Then, this score is used as a feature to train the classifier.

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. The way we currently define an entity mentions is if a) the name fully appears in a sentence b) one part of the name appears in a sentence. A tf-idf score of a wiki page against the sentence that this name occurred in can be used for disambiguating names in the book. But, for this submission, we only consider scenario b, where appearing one part of a name is enough for being considered as an entity occurrence.

Evaluation

In this section, we evaluate the project and analyze the obtained results. The metric we used for evaluation is F1-score. In total we have five sets of features; 1) total mention count 2) mention count per book 3) gender (frequency of pronoun and closest pronoun) 4) number of occurrence with a deadly name at the same sentence 5) tf-idf of collection of deadly sentences. The following plot is F1-score of various classifiers using the same set of features.

As one can see in figure 1, decision tree has the best score, with the f1-score of 0.401. One thing that is obvious from figure 1 is that linear models didn't work well at all. They were not able to generalize the data.

In another experiment, the rule-based gender classifier explained in section Approaches trained and tested using the same set of data as the previous experiment. In this experiment, at first a frequency based model is tested. This model could obtain a f1-score of 0.744. 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 classifiers. The result of this experiment is shown in figure 2.

It is very interesting that using any type of classifier and these features, we can obtain a high f1-score of 0.9. The type of classifier did not make a huge difference in this case.

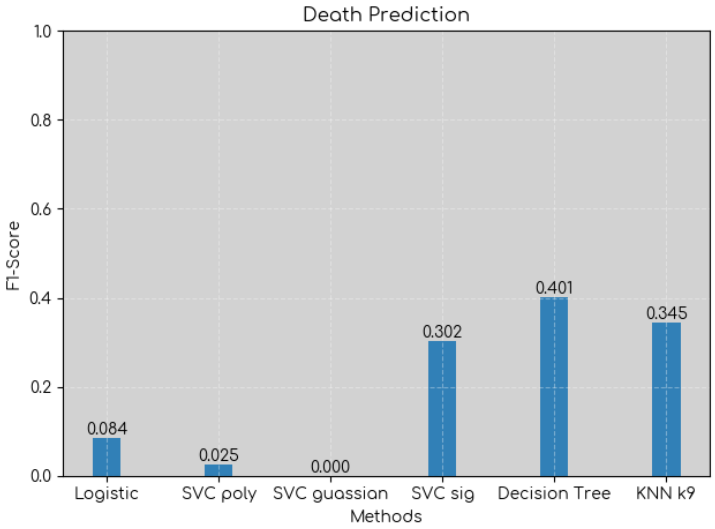

Figure 1: F1-score of various classifiers on a set of features completely obtained from the text.

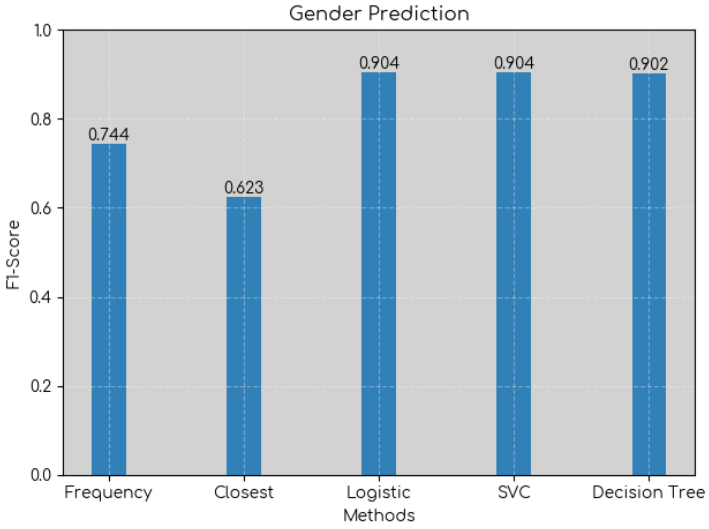

Figure 2: F1-score of gender classifier. The first two bars are related to the rule-base predictor, and bar 3-5 are f1-score of different classifiers using those rule-base feature for training.

Future Work

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.