Superheroes: Web-Scraping and Natural Language Processing (NLP) Applied to Predictive Modeling in Python

Daniel Maiorano, Justin Moczadlo, Grace Seaton, Aaron St. John, Reilly Williams

***Abstract*— This project scraped the SuperheroDB website to compile metadata for all provided superheroes. The raw data was processed, with the text attributes going through a natural language processing pipeline. The cleaned text columns were then used in Random Forest and XGBoost models to predict the creator of a superhero from the compiled background history text. The best performing model was the Random Forest model, with a prediction accuracy of 81.4%.**

I. OBJECTIVE

Have you ever thought to yourself just who is strongest superhero? Or, why every superhero's parents died mysteriously? Maybe you want to figure out if having dark-hair and being tall and brooding always means that a character will be an antihero? Maybe you just want to see the correlation between regenerative abilities and how many times a hero has died? Perhaps you don't know many superheroes, but you wonder what might be the strongest super power to posses. Well the objective of this project is to dive deep into a database of superhero information. Of particular interest to us is using the text columns in the scraped dataset to classify the creator of the superhero (Marvel, DC, other).

The motivation for this project came from our nerdy tendencies and interest in all things related to superheroes. We also didn't know an entire database of superhero data existed. This will be a great opportunity to explore the data and find interesting insights.

II. DATA PROCESSING

The data for this project comes from the website SuperheroDB.com (SHDb). SHDb is similar to Wikipedia as anyone can edit the information for a given superhero. There is no official "approval" process for edits, instead it relies on its users to correct any incorrect information. Currently, there are over 28k superheros and villains on the site.

A. Data Collection

Initially, we planned on using a compiled dataset scraped from this website and offered open-source on Kaggle, however, we decided to instead recreate the original dataset creator's process and re-scrape the data ourselves to update it and keep it current as the original data hasn't been updated for over two years.

To compile a more recent version of the dataset, we wrote a python script that found all the unique superheroes on SHDb and output a shell script that could be run to get each hero's About, History, and Powers pages. The final script to collect all of these web pages wound up around 115k lines long and took 6 days to run. The output of this script was around 63k html web pages that were then processed to extract useful information for each superhero.

In more detail, we iterated through the pages of characters in SHDb and collected each unique character that was listed. Then, for each character, we generated three links: '_about.html', '_history.html', and '_powers.html'. These links for each hero were then concatenated in a script that downloaded those web pages for each hero to a folder in our repository.

B. Data Processing and Feature Engineering

After the raw web pages were collected, they were cleaned using BeautifulSoup and re (regular expressions). From the 'about' web pages for each hero, we extracted the characters image, real name, overall score, power stats, super powers, origin, connections (what team they may be on, etc.), and appearance descriptions (eye color, height, hair color). From the 'history' web pages, the history text field was extracted to be later processed using natural language processing (NLP) techniques. Lastly, from the 'powers' web pages, the particular powers, including powers subtitles and content, were extracted for the heroes.

After using BeautifulSoup and the raw files to generate those above mentioned features, more feature engineering work was done to enhance the dataset. Some generated features include what teams a hero is on, exploding the overall score column into intelligence, strength, speed, durability, power, and combat scores, along with generated 50 dummy variable columns for the top 50 most frequently occurring superpowers.

C. Natural Language Processing

Since the objective of this project was to predict creator from the text columns, we had to employ some natural language processing (NLP) techniques to transform the 'history' and 'powers' text columns into usable states for machine learning algorithms.

The first step in this process was to fill any null text columns with a blank space so that the other steps in the pipeline would not produce errors. After filling the nulls, the text was all converted to lowercase. Next, stopwords were removed from the text. A stopword is a word that offers little or no meaningful information such as "the", "a", "up", etc. We used the built in stopwords module from nltk to remove all common English language stopwords. Lastly, the text was lemmatized (all verbs were truncated to their root form) and tokenized (sentences were converted to lists of unique words by splitting on whitespaces).

D. Final Dataset

After all of the preprocessing steps, our final dataset had 1238 heroes, and 81 columns, Excluding the 50 dummy variable columns, we're left with 'name', 'real_name', 'full_name', 'overall_score', 'history_text', 'powers_text', 'intelligence_score', 'strength_score', 'speed_score', 'durability_score', 'power_score', 'combat_score', 'superpowers', 'alter_egos', 'aliases', 'place_of_birth', 'first_appearance', 'creator', 'alignment', 'occupation', 'base', 'teams', 'relatives', 'gender', 'type_race', 'height', 'weight', 'eye_color', 'hair_color', 'skin_color', and 'img'.

Our primary columns of interest are 'full_name', 'history_text', and 'powers_text'. The 'full_name' column simply contains the full name of the super hero. The 'history_text' column the cleaned tokenized string column that details a hero's backstory. Finally, the 'powers_text' column is also a tokenized string column detailing a hero's superpowers.

III. PREDICTIVE MODELING

As mentioned above, the objective of this project is to build a predictive model for creator based on the cleaned history text column for each superhero. To accomplish this, we decided to test two models: random forest, and XGBoost.

A. Random Forest

Random Forest is an easy to understand ensemble algorithm that can be applied to both regression and classification tasks. This algorithm builds several uncorrelated decision trees and combines them by using their average to then decide on a more accurate classification of our data, in our case, classifying superhero's creator.

We converted our text features, history_text, into numerical by using TFIDF (Text frequency inverse document frequency), limiting our max features to 10,000 and using bigrams and unigrams, and nltk's stopwords list. This resulted in us having 10,000 features that were either one word or two words and normalized. We only integer encoded our labels (0, 1, 2, 3...), though they could be left as categorical, since one-hot encoding random forests can introduce sparsity into our features and degrades the performance. Using GridSearchCV from sklearn.modelselection we decided to do some rudimentary tuning of three hyper parameters for the random forest: n_estimators, max_features and criterion. N_estimators is the number of decision trees we want to build before taking the averages of the predictions and we tested values ranging from 100 to 1000. Max_features is the maximum number of features it considers when building a tree and we tested the log(number of features) and sqrt(number of features). Lastly, criterion is the type of purity measure the algorithm uses to measure information in a decision tree split, either gini or entropy. The accuracy of our model after tuning the three chosen parameters was 81.4%, with the n_estimators set to 700, sqrt max_features and using gini for our criterion.

B. XGBoost

XGBoost is an algorithm that utilizes gradient boosted decision trees, and excels at speed and performance for structured or tabular data. Boosting is an ensemble technique where a new model is added to correct the shortcomings of the current models. The XGBoost algorithm continues to add models until there are no more shortcomings in the ensemble. The XGBoost package in python offers many ways to tune the model so one can easily find the best one. However, for our project, the group wanted to compare XGBoost to a Random Forest to see which performed better. As such, we did not regularize the data and we simply used 200 estimators in the model. However, before we ran the models we had to transform the input and output data. We one-hot encoded our labels, which in the context of the assignment was the creators of the comic (DC comics, Marvel comics, etc.). We normalized the history text by passing it through a TF-IDF normalization function. Once the inputs and outputs were in numerical format we were able to pass it into the XGBoost algorithm to predict the creators of the comic. The resulting accuracy of our model was 73.9%.

IV. ISSUES ENCOUNTERED

This project was very fulfilling to work on, and scratched a nerdy itch in all of us, but there were several difficulties that we encountered along the way.

First, the script written to scrape SHDb took over 6 days to run. The sheer number of characters/superheroes/villains contained on that website far exceeded any of our imaginations, and also far exceeded the computer power of our personal laptops. The time spent to run the script was a major blocker in our project, as almost all work had to stall for one week until the data was collected. This led to a rush at the end of the project life cycle to complete our models, and contributed to a lack of appropriate time spent exploring our data before we fit the models.

Yet, in spite taking 6 days to scrape the website, the majority of the scraped web pages were junk. Initially, there were around 64k unique characters that were compiled, however, after extracting the features we wanted for our data and after the feature engineering work, most all of those characters were removed because they either did not have history text (the column we were modeling on) or they were completely blank except for a name. In the end, less than 1300 of the original 64,000 characters were retained for training our models. That is a 98% loss rate for our data, and was very disappointing to see. We were fortunate that we still had enough data to train our models, but if we had lost much more, we would have had to employ bootstrapping or other sampling methods to artificially increase the size of our training data.

All in all, we enjoyed the project, but the data quality was very poor, and it required far more extensive cleaning than we initially anticipated.

V. FUTURE WORK

Despite the high performing models that we built, we still strongly believe that more could be done to further enhance the performance of our models. If we had all the time in the world, we would like to try the following:

- Bootstrap data sampling: Bootstrap sampling is sampling with replacement. It is often used in machine learning to estimate statistics on a population, but can also be used to increase the size of a testing or training dataset without having to create entirely dummy data. We would have liked to use this in the random forest and XGBoost models to increase the training size, but also to validate the performance of both models.
- Dimension reduction techniques: Dimension reduction is the transformation of data from a high dimensional space to a lower dimensional space while retaining the most meaningful properties of the data. In the context of NLP, dimension reduction is often used to reduce the number of words contained in a dataset to only those most important in a model. Two popular dimension reduction techniques we would have liked to have tried on this data are:
  - Latent Dirichlet Allocation (LDA): LDA is a dimensionality reduction technique that is frequently used in NLP problems. Since it is a dimensionality reduction technique it tries to eliminate as much information as possible while retaining the most useful information. This is something we could explore in future iterations of the project.
  - Principal Components Analysis (PCA): PCA is another dimensionality reduction technique but works on matrices instead of text like LDA. Since we one-hot encoded our data in some of the models, this could be explored to see how it improves or speeds up the model.
- Further data exploration: There were two avenues we explored to mitigate our lack of data. The first, as previously mentioned, was re-scraping the web pages, which unfortunately led to junk data. The other avenue we looked into but deemed too complex and too foreign to the group was data augmentation. The group has used data augmentation in the past, but for numerical data and for images. The group was fearful to augment text data, as we were unsure if we would do it correctly. We did not want to generate more data at the cost of data integrity. However, if someone were to pick up where we left off this would be a natural path forward.
- Further model exploration: The group wanted to explore deep learning techniques. Each of us were in a deep learning class and learned how to tune and modify deep learning models to get the best performance out of them. Using deep learning in an NLP context would have been very fulfilling, however we were limited by the size of our data. In the future if the data issue was resolved it would be nice to compare a cutting edge technique like deep learning, to some current best in class techniques like random forests and XGBoost algorithms.

VI. RESULTS

This project scraped metadata about every superhero that was listed on SuperheroDB (SHDb). The raw data was then transformed into a useable state, and the text columns from the outputted dataframe of those transformations were run through a basic natural language processing (NLP) pipeline to extract as much information as possible from the text. Unfortunately, during the cleaning process to transform the data into a usable state, we lost about 98% of the raw data due to the large amount of null values and other data quality control issues. Nevertheless, we still have over 1300 heroes with full data to work with.

As the goal of this project was to see if we could predict the creator (i.e. Marvel, DC Comics) of a superhero based only on the text columns, we then decided to fit two models, a random forest model and an XGBoost model. The best of these two models was the random forest model with an accuracy of 81.4%. This means that we can accurately predict the creator of a given hero with an accuracy of about 4/5. Obviously, this may be because there is a large preference in the data to Marvel or DC Comics, which make up about 80% of the data, but the confusion matrices seen in our presentation show promising distinctions between those two creators, indicating that the models did discern different variation in the features for those major creators.

There are other things we wish we could have done with the data and with the models, but nonetheless this preliminary exploration was a gentle introduction into the predictive power of the dataset that we created.

VII. TEAM MEMBER RESPONSIBILITIES

Each of the five team members working on this project brought their own unique skills to the table when scraping, cleaning, and modeling the data.

- Daniel Maiorano: Worked on the XGBoost analysis and wrote the XGBoost section of the paper as well as contributed to future steps.
- Justin Moczadlo: Worked on RandomForest analysis, reviewed RandomForest section of the paper. Also contributed to the RandomForest and WordClouds portion of the slides.
- Grace Seaton: Worked on RandomForest analysis, wrote RandomForest section of the paper, and also contributed to the slides.
- Aaron St. John: Worked on the XGBoost analysis as well as reviewed the XGBoost portion of the paper. Also contributed to the XGBoost portion of the slide as well as future steps.
- Reilly Williams: Reilly was responsible for the bookends of the project. For the beginning of the project she completed the Data Processing steps, including scraping the dataset, extracting the information from the raw web pages, feature engineering, and the NLP pipeline. Reilly was also responsible for the final leg of the project, including our model evaluations and comparisons, as

well as compiling our findings. Lastly, she created the final PowerPoint presentation for the class presentation.