**Agent Prediction in Narrative Texts**

**Participants**

Ross Newman [rossnewman@my.unt.edu](mailto:rossnewman@my.unt.edu) rcnewman

Mica Haney [mrowanh21@gmail.com](mailto:mrowanh21@gmail.com) HamsterHerder

Dhana Sanketha [sanketharepala@gmail.com](mailto:sanketharepala@gmail.com) sanketharepala

Varunraj [varunraj1816@gmail.com](mailto:varunraj1816@gmail.com) varunraj1816

**Abstract**

The project Agent Prediction in Narrative Texts seeks to build an architecture that can detect both the protagonist and the antagonist of a narrative in text form. The ultimate goal of the project is to describe the intensity and classification of the relationship between the protagonist and antagonist, but the participants of the project are aware that this goal is complex and possibly not achievable in the available time.

The individual characters will be analyzed based on the words that occur in reference to each character. This analysis will generate a classification of what type of agent each character is, and how they interact with the other characters. This network of characters and interaction can then be analyzed to detect the protagonist and antagonist. These will be the agents with the greatest influence over the narrative and most intense negative relationship between them.

**Related Projects**

Characterizing Social Relations Via NLP-based Sentiment Analysis

Georg Groh, Jan Hauffa

<https://pdfs.semanticscholar.org/2005/7ff308e2fd196feb64775fbd70d2fbb44a0c.pdf>

Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized: On The Difficulty of Detecting Characters in Literary Texts

Hardik Vala, David Jurgens, Andrew Piper, Derek Ruths

<https://www.aclweb.org/anthology/D15-1088.pdf>

**Workflow**

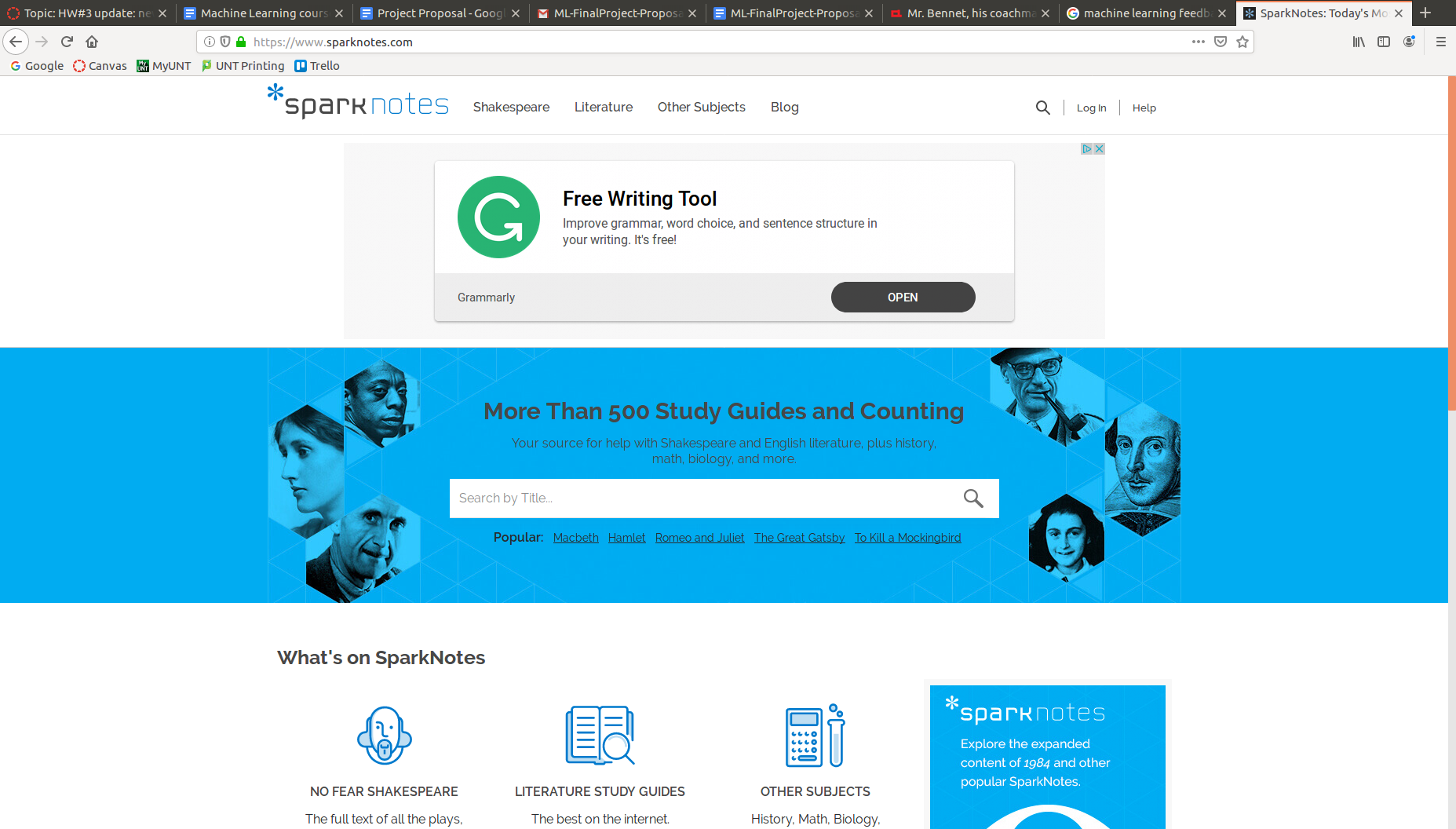
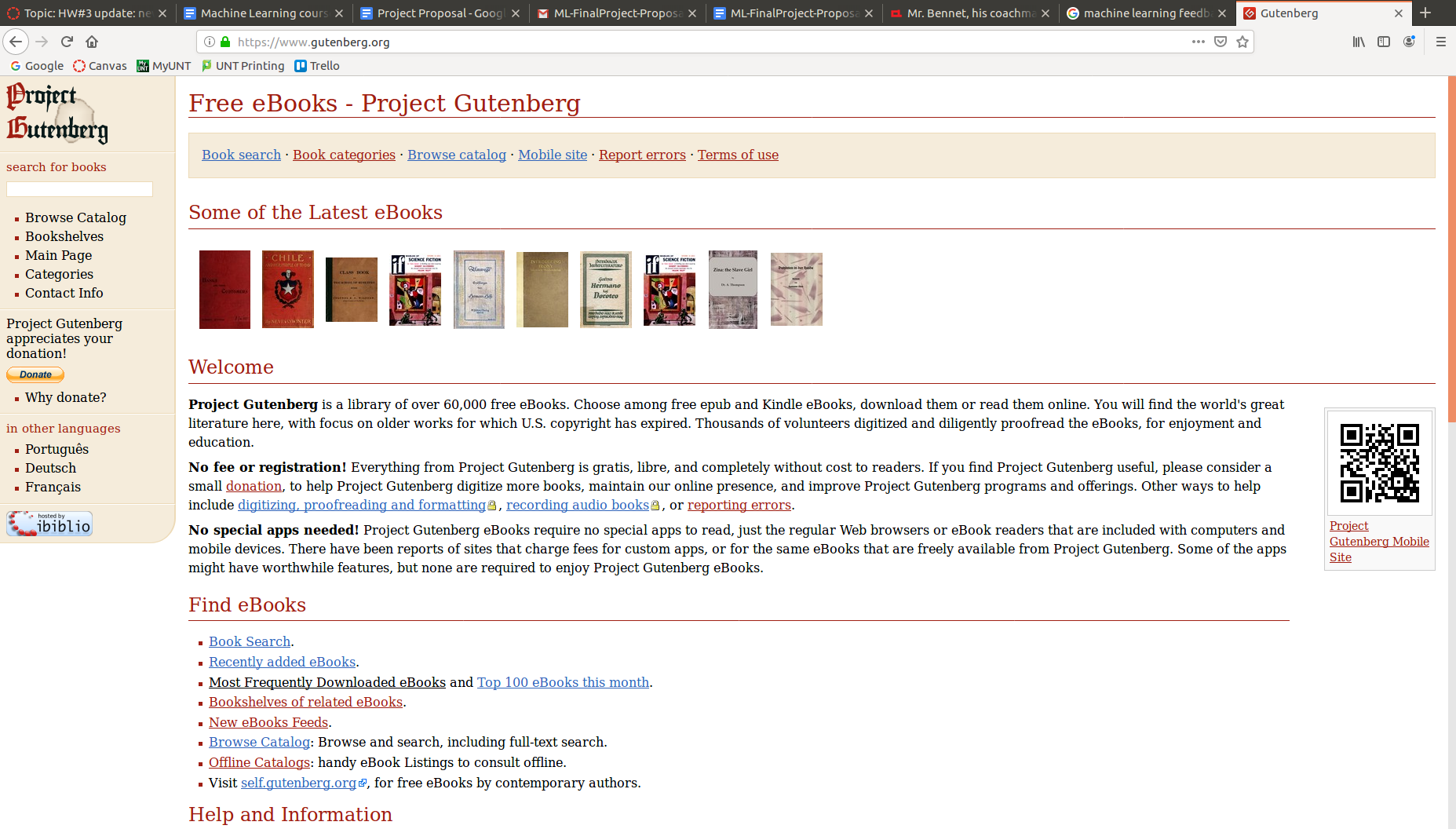
All coding work has been committed to a Github repository. Project tasks have been documented and assigned via a Kanban board on Github. Any urgent concerns have been handled via a group text message.

*Github:* [*https://github.com/rcnewman/agent-prediction-ml-finalproject.git*](https://github.com/rcnewman/agent-prediction-ml-finalproject.git)

*Kanban:* [*https://github.com/rcnewman/agent-prediction-ml-finalproject/projects*](https://github.com/rcnewman/agent-prediction-ml-finalproject/projects)

**Dataset**

The data was retrieved from the websites of Project Gutenberg and SparkNotes. The plain documents were retrieved from Project Gutenberg. Initially this was to be done via data scraping. However, the code written to do this resulted in IP banning. This led to the data being downloaded by hand. The protagonist and antagonist were retrieved from SparkNotes, again by hand.



(a) (b)

Figure 1: (a) A screenshot of the homepage of Project Gutenberg (<https://www.gutenberg.org/>). (b) A screenshot of the homepage of SparkNotes (<https://www.sparknotes.com/>).

The data was transformed, by document, into a matrix. Each row of the matrix represented a named entity in the document. The first column is the relative counts of the number of times the entity was referenced. The second column is the sentiment score for the entity, ranging from -1 to 1, and averaged across the entire book. The features are described further in the Project Design section.

The data was split into training and testing sets, at a roughly 80-20 ratio. The data was not randomized for this selection.

**Project Design**

Due to the inherent unstructured nature of natural text, most NLP tasks require features to be generated from the data. Our task requires this, as our data is completely unannotated.

Each document is first tokenized by sentence, and each sentence is tokenized by word. Each tokenized sentence has the part-of-speech tags added, and then named entity extraction is performed. These actions are done using nltk’s packages. All named entities that are people or organizations are added to a dictionary, with methods in place to make sure that there are no repeat entities are added.

For each sentence, a sentiment score is generated using nltk’s vader plugin. These scores are added together for each named entity encountered, and are later averaged across the entire document. This generates an average sentiment for the named entity across the entire document. The score assigned ranges from -1 (most negative) and 1(most positive). Vader is a pre-trained sentiment analysis model, trained specifically over social media. This is admittedly a poor fit for the data. There are two reasons vader was chosen. The first is that it’s easy to use, due to nltk’s plugin. It was easy to start with, and could be replaced later if there was time to do so. The second reason is that it was one of the few pre-trained python sentiment analysis tools available.

After all the sentences in a paragraph had been evaluated, the paragraph itself was analyzed for coreference resolution. It was decided to do coreference resolution on the paragraph level rather than the sentence level because a character can easily be referenced by name only once in a paragraph, but be discussed multiple times by pronouns. Originally coreference resolution across the entire document was attempted, but the process was deemed to take far too long and produced no better results than merged paragraph level results. The final coreferences were put into a dictionary, with methods in place to combine duplicate entities. The coreferences were then counted. The relative number of times an entity appeared in the document was computed. The coreference resolution was performed using the pre-trained spacy and neural coref models, both of which are trained on English models. Spacy and neuralcoref were chosen for - again - the ease of use and the fact that they are pre-trained.

Our baseline model was simply looking at the coreference counts, and assigning the protagonist as the highest count and the antagonist as the second highest count.

To evaluate the documents for the protagonist and antagonist, the features for each document were converted to a matrix. The first column was the name of the entity, the second was the relative coreference count, and the third was the average sentiment score. The gold labels were generated and placed into a list by evaluating if any of the entities matched the provided protagonist and antagonist. For training, all of these matrices and lists were concatenated and fed to the various models that were run. For prediction and evaluation, the matrix for a single document was fed to the trained model, and the entity with the best predicted label probability for the label was selected. Since evaluation for protagonist and antagonist were done independent of each other, it is possible that the same entity could be selected for both. This is desirable, since for multiple documents in the dataset the protagonist and antagonist were the same person.

Four models were trained and evaluated, using Sci-Kit Learn’s implementations. The models trained were: SVM, Naive Bayes, Multilayer Perceptron, and Decision Tree.

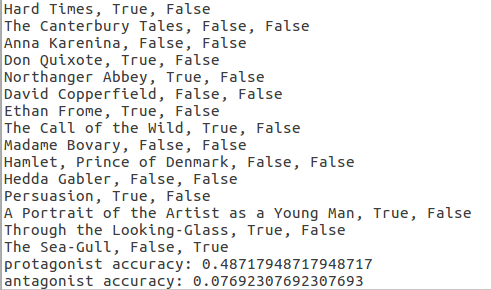
After initial results, hyperparameter tuning was run on the SVM and Multilayer Perceptron. The Naive Bayes and Decision Tree did not have the right kind of parameters to effectively leverage the data.

As a final experiment, adaboost was applied to the data.

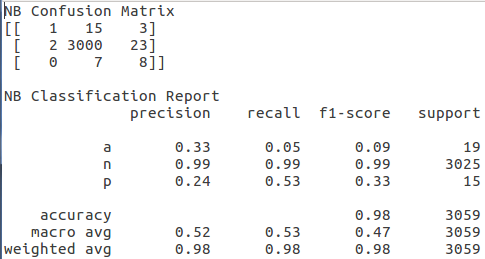
As expected, there was not time to implement relationship analysis in this project.

**Results**

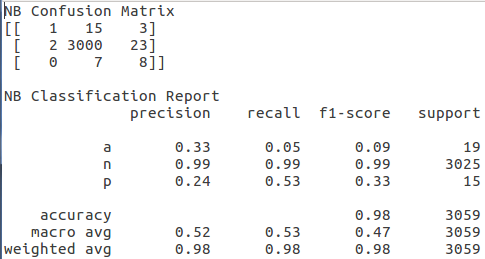
Baseline:



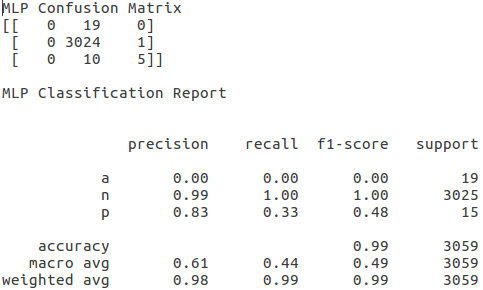
SVM:



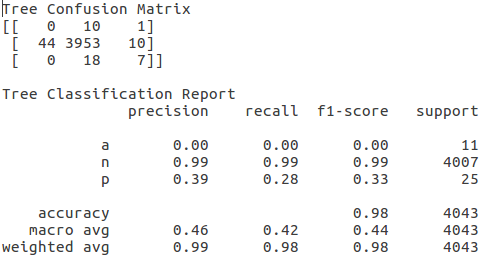
Naive Bayes:



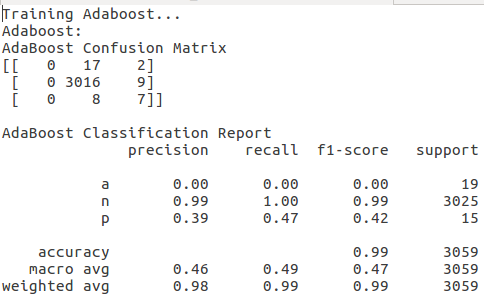
Multilayer Perceptron:



Tree:



Adaboost:



**Milestones**

The milestones and who has performed the task are listed below.

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Description** | **Assignee** |
| Data Collection | Data scraping of Project Gutemberg and labeled via scraping of SparkNotes. | Ross Newman, Dhana Sanketha, Varunraj |
| NER | Get Named Entities via NLTK. | Ross Newman, Mica Haney |
| Classification of Named Entities | Classify Named Entities to extract people from non-actor entities (character detection). Will be done using NLTK’s Stanford CoreNLP plugin to allow for use of CoreNLP’s NER Classifier. | Ross Newman, Mica Haney |
| Character-Level Coreference Resolution | Use coreference resolution to link mentions of the same character together. Use this and potentially an occurrence count to detect protagonist and antagonist via assuming them to be the most referenced characters in the document. This assumption may be revisited at a later point. | Ross Newman, Mica Haney |
| Character-Level Sentiment Analysis | Analyze sentiment attached a single character to detect whether the charter is in an antagonistic role or not with the intention of detecting the protagonist and antagonist. | Ross Newman, Mica Haney |