CS 410 Project Progress Report: Leaderboard Competition for Detecting Sarcasm in News Headlines and Articles

Greg Lee (gglee2) [Captain]

1 Problem Definition

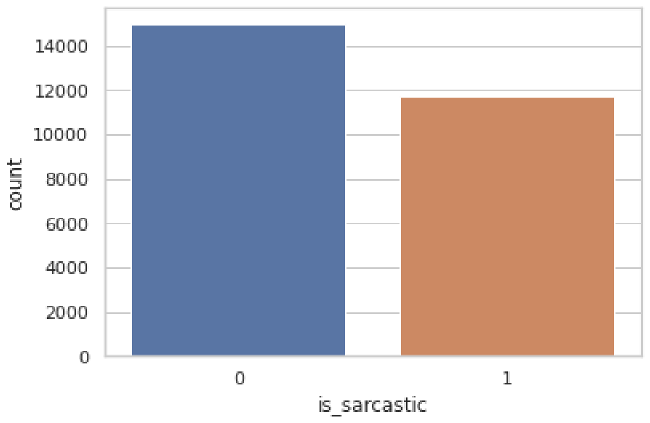
My project is a Leaderboard Competition hosted on LiveDataLab. The task that participants are solving is training a model which predicts whether a new article is sarcastic based on its headline and contents. In other words, this is a binary classification task: given a news article, classify it as "Sarcastic" or "Not Sarcastic". This is an important problem because many people consume news at a glance and may unintentionally spread misinformation if they don't realize an article is satirical. Additionally, for web scraping tools that aggregate news, it would be extremely useful to have a tool which can automatically detect (and filter out) articles which are not intended to be serious news.

2 Progress

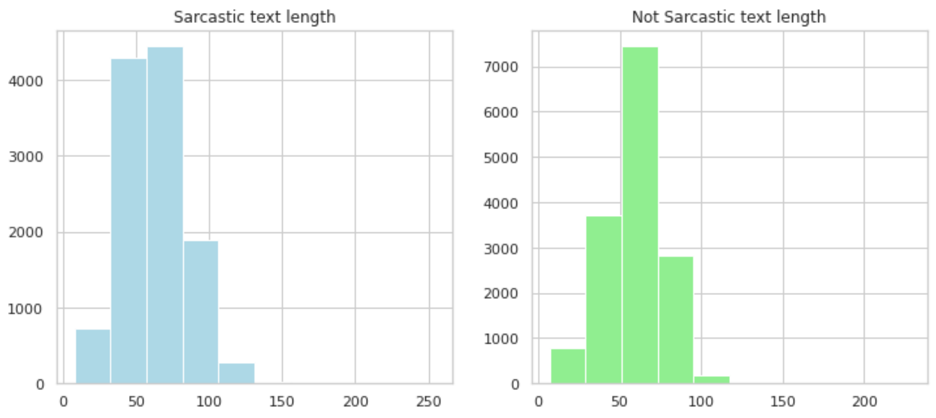
Dataset and Exploratory Data Analysis

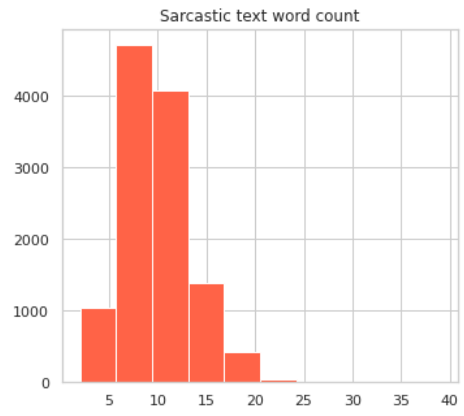
The first step taken for creating the leaderboard competition is actually downloading the dataset and performing exploratory data analysis to understand its shape and the specific features available to the people participating in the competition. Here are some of the basic findings:

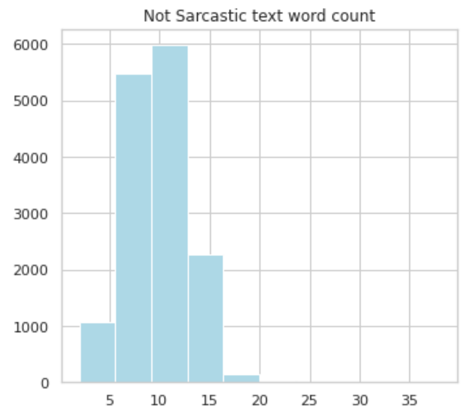
There are a total of 28,619 articles in the dataset, each with a hyperlink to the article, the article's headline, and whether the article is satirical or not. In the dataset, 14,985 are genuine news articles and 13,634 are satirical news articles. This implies that (based on our dataset), the prior probability of any given article being sarcastic is about fifty-fifty.



Additionally, I looked into the length of these headlines (in terms of both character length and word length). This is important because if a participant decided to use something similar to TF-IDF weighting like we've discussed in class, it's good to know the lengths of these article headlines and if they're similar across both satire and non-satire. From the results, I observed that both types of news articles have similar lengths but non-sarcastic articles tend to be longer. This makes sense since a "real" article would have more information it needs to get across in the headline. However, their lengths are overall pretty comparable.

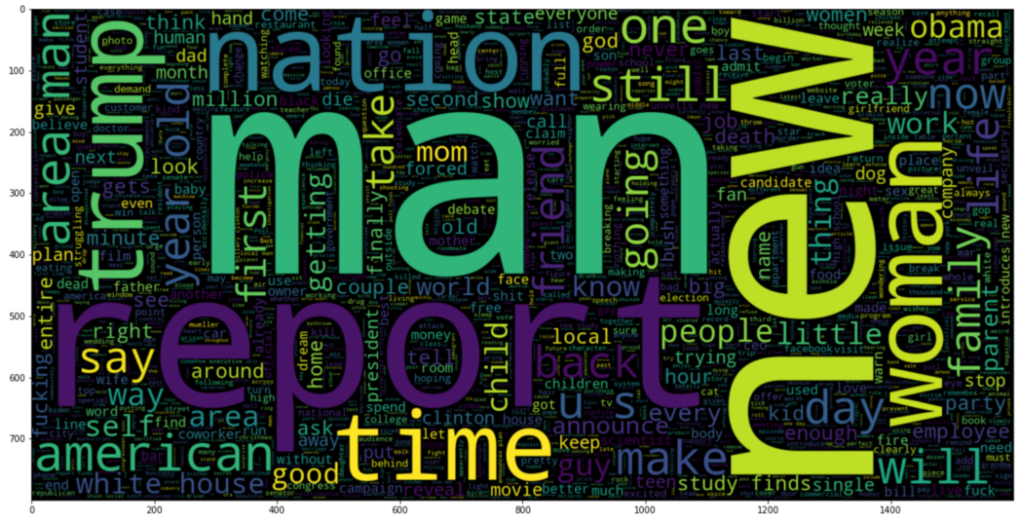




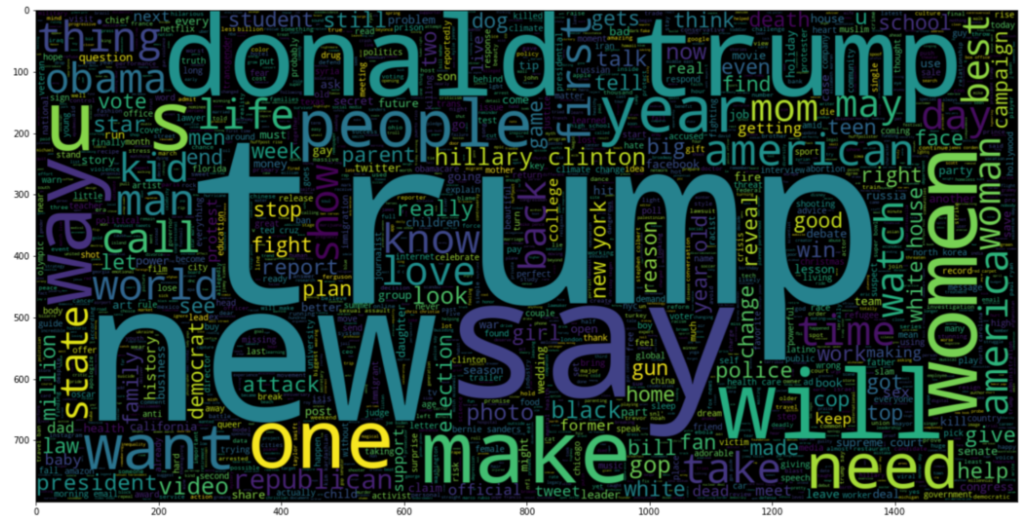


The last step of Exploratory Data Analysis (which admittedly was mostly just for fun) was generating a word cloud to quickly spot check which words occur most frequently in sarcastic news headlines vs non-sarcastic ones.

Sarcastic Headlines



Non-Sarcastic Headlines



From these word clouds, one obvious indicator of a headline being satirical seems to be the presence of the word "man" or "woman" to describe a general (fictional) person rather than the use of real names. Another interesting observation is that the topic of Donald Trump, the US president during the time that these news articles were written, is common in all types of news articles. Quantitatively, the top five words for sarcastic headlines are: man, new, report, area, and nation. For headlines which are not sarcastic, they are: trump, new, donald, says, and women.

Simple Baseline

After getting a basic understanding of the data and splitting it into training and testing datasets, I developed a simple baseline. Before even implementing any specific models, I preprocessed the data to manually remove punctuation and used the ntlk package to remove stopwords. With the data preprocessed into a discrete form, I implemented the baseline. It's worth noting that for the purposes of understanding the perspective of a student participating in the LiveDataLab competition, I implemented my classifier and evaluation methods manually. In reality for the competition, I would use sklearn's prebuilt classifiers since they're probably better than mine.

The simplest model I could think of was one that did Naive Bayes on the article headlines in order to classify them as 1 (Satirical) or 0 (Not satirical). The Naive Bayes classifier simply constructs the conditional probability that an article title is satirical using Bayes' Rule. Namely, it tries to model the posterior probability $P(Satire|Words)$ using the likelihood $P(Words|Satire)$ and the prior probability $P(Words)$.

The prior probability is simple to calculate (in fact, we already did it in the exploratory data analysis) and it's counting the proportion of articles that are satirical vs non-satirical. To compute the word likelihoods, we use the conditional probabilities of each unigram word. AKA for each word, count up how often it appears in satirical headlines and how often it appears in non-satirical headlines. Then to compute the likelihood of an entire headline, we compute the product of each individual word's probability. We don't need to do the logarithm trick to prevent underflow because the length of these article headlines is low. Ultimately, the prediction is made by choosing the class with higher posterior probability.

I also implemented the three evaluation metrics: Logistic Loss, Accuracy, and F1-Score. And to verify both my baseline model and my evaluation implementations, I evaluated the my model's performance on the test dataset. The results are as follows:

Evaluation Metric	Score
Baseline Log Loss	0.351
Baseline Accuracy	0.768
Baseline F1-Score	0.732

Clearly, the baseline results aren't perfect since it is one of the simplest possible models but they are reasonably good. This should be a good way for participants to quantitatively see how much their techniques improve upon this existing naive baseline.

3 Tasks Left to Do

Next Steps

The immediate next steps to take for the project are to write the code to integrate the existing evaluation metrics and the data with the LiveDataLab leaderboard. This would involve reaching out to TAs (via CampusWire) to ask for some tips on working with LiveDataLab and hopefully finding documentation. (Note: Since initially writing this progress report, a TA has reached out to me with helpful resources on LiveDataLab's interface, thanks!) In specific, I need to write code which takes in the student's Python file, runs it on the training dataset, evaluates on the test dataset, and then displays those results on the leaderboard.

After implementing that leaderboard, I will also need to create formal documentation for the code, prepare for the presentation, and work on the final project report. Looking at my original timeline, it seems that I have completed the goals I set out to finish by this progress report. Namely, I've done exploratory data analysis and implemented a simple baseline with evaluation metrics. So if I keep the same pace on the project, I'm looking on track to complete it.

Timeline

Task	Due Date
Submit Progress Report/Team Formation	10/22
Exploratory Data Analysis	10/26
Develop Simple Baseline	11/4
Implement Evaluation Methods	11/10
Submit Progress Report	11/14
Integrate Code with LiveDataLab	11/21
Finalize Documentation	11/25
Presentation	11/30
Submit Final Project	12/7

4 Challenges Faced

One challenge for me with this project was dealing with the format of the data from the Kaggle dataset. This is in JSON (Javascript Object Notation) and I'm not familiar with Javascript. So something that I had to do during my exploratory data analysis was converting from a JSON-type object to a Python-type object that could actually be used in a machine learning model. Originally, I actually wrote manual code to process these JSON objects but then I learned that Pandas is able to read them into a dataframe.

Another challenge I faced with creating the baseline was actually something that we discussed in the course: the problem of smoothing. My initial baseline involved vectorizing the documents using count but that led to some 0 probabilities for words that appear in the test set but not in the training dataset. I resolved this using Laplace smoothing, adding a small constant for all words to prevent these zero probabilities.

The most significant challenge for me on this project was understanding LiveDataLab itself. I was hoping for feedback from a TA that could point me toward resources to help but I started most of the work before feedback came through. I read through the Leaderboard section of Aaron Green's Thesis on LiveDataLab (https://www.ideals.illinois.edu/items/115630) but was unable to find any concrete technical details to help me implement the backend of a leaderboard. As a result, I focused mostly on understanding the data and developing a baseline classifier for this progress report. (Note: I now have access to resources about LiveDataLab integration and that should really help with this challenge).