“Can we reliably detect sarcasm in a comment?”

David Kaiser, PhD
Fundamentals of Data Science, Spring 2020

INTRODUCTION:

Sarcasm is hard to define, as we see in the four divergent dictionary definitions below:

- Merriam-Webster Dictionary: "a sharp and often satirical or ironic utterance designed to cut or give pain"
- Oxford Dictionary: "the use of irony to mock or convey contempt."
- Cambridge Dictionary: "remarks that mean the opposite of what they say, made to criticize someone or something in a way that is amusing to others but annoying to the person criticized"
- dictionary.com "harsh or bitter derision or irony."

As hard as it can be to define it, it can be even harder to spot it. In spoken conversation, sarcasm relies heavily upon tone, inflection, and facial expression or other gestures. All of these are missing in text, although text can be enhanced with emojis, punctuation or other markers. As a result, it can be very difficult for a reader to pick up on the writer’s intention.

But perhaps it is detectable, based upon features of the text. Perhaps we just haven’t found the correct features or combinations of features.

In this study, we will use machine learning techniques to classify Reddit comments as either Sarcastic or not.

DATA:

The data file from Kaggle (see appendix for URL) is a .CSV format which is more than 250MG.

The data set contains 1,010,826 comments, evenly split into those which were labeled sarcastic (by users, using the '/s' tag) and those which were not. Those which are sarcastic have the "label" value of "1" in the data set, those which are not have label "0." We will be trying to predict if a comment should be labeled 0 or 1.

The data set has ten columns, including the label (boolean), the comment itself (text), the author's username(text), the total upvotes (int), downvotes (int) and net (upvotes minus downvotes, int), the title of the subreddit where it was found (text), the month / year of creation (date), the UTC timestamp of creation time (date-time), and the parent comment it was posted to (text).

The good news is that the data appears to be complete, with no NA values that need to be cleaned or deleted.

The bad news is that the data set is so big, it would consistently hang up during creation of the model. As a result, we decided to use a sample of 5 percent of the total, selected at random. Because it is random, it remains fairly representative of the whole. Testing with 10 percent and 20 percent samples did not display significant improvement, and using the smaller sample sizes allowed for frequent testing over a variety of models.

GOAL:

In this project, we attempt to predict sarcasm from the text of the comment alone, we won't be incorporating upvotes or downvotes, the subreddit, etc, although it is certainly possible that these may improve the model by inclusion. Even though we are only using one field, we will face some difficult decisions regarding text preprocessing.

While many text-related tasks typically put all of the letters into lower case and remove punctuation and non-word characters, and then possibly run the words through a stemmer and / or lemmatizer to standardize them, it

may be the case that these “non-sterilized features” are necessary to predict sarcasm, so we may need to keep them. We tested a variety of text processing functions and will report on the results.

METHODOLOGY:

We started by randomly assigning 75% of the sampled corpus of 50,541 comments to a training set, and holding the remaining 25% as a test set. We used the libraries text2vec and data.table for processing the text, using a simple bag of words model. Although a bag of words may be considered simplistic, it has proven surprisingly functional and robust in practice.

The word2vec libraries requires that we set a tokenizing function and a preparatory function to be used prior to input into the function that creates a matrix of tokens from which vectors and a model will be generated. The preparatory function was the source of variation over the different models.

Model One used only the “tolower” function, which changes all uppercase letters to lowercase.

Model Two used “removePunctuation,” which removes all of the various punctuation marks (for example: !, ?, “, ‘, etc).

Model There used “removeNumber,” which replaces all of the digits in the text with an empty string.

Model Four used “removeWords” to removed the English language “stop words” (according to the R text-mining library “tm”). Stop words are the common grammatical words that don’t contribute much to the semantics of the sentencee, for example, “the,” “and,” etc.

Lastly, Model Five used a composite function called “doItAll,” which put all text into lower case, and removed punctuation, whitespace, numbers and stop words.

The output of these pre-processing tasks is then input into a tokenizer, then a vocabulary is created, and this is used to create a document term matrix (DTM). A DTM is a matrix which lists the terms that are present in each “document,” here, a Reddit comment. Lastly, this matrix is used to train the

classifier. We use the cv.glmnet classifier from the R library “glmnet,” with the algorithm set to “binomial.” The resulting output is a predicted “value” of the logistic regression, with the default being a split at .50 (below is “negative,” above is “positive”), but this can easily be set higher or lower based upon our understanding of the data and the costs and benefits associated with the trade-off between specificity and sensitivity. We then evaluate the classifier based upon how well it can predict the outcome of the training corpus.

Lastly, we evaluate the model by using it to predict the output of the documents in the test sample which was held back previously. This speaks to the generalizability of our model. The best model in the world for the training data is useless if it can’t be used for novel data in the “real world.”

The methodology for evaluation of the various models was the Area Under Curve for a Receiver Operating Characteristic curve (ROC curve). The ROC curve is useful here because it gives us an overall impression of a model, while allowing for the possibility that we may want to change how we assign the reported outcomes, as discussed in previously, in accordance with our understanding of what a well-trained model will render. The evaluation score ranges from 0 (never correct) to 1 (always correct).

As an aside, we use set.seed to ensure that the results are consistent across multiple runs of this program.

RESULTS:

Below are the results, as measure by the max AUC on the training corpus, for the five models discussed above.

Model 1 (lower only)	0.7140852
Model 2 (punctuation)	0.7157454
Model 3 (numbers)	0.7170152
Model 4 (remove stop words)	0.7063882
Model 5 (doItAll)	0.7109954

All of the models scored in the same vicinity, predicting the correct result 70 - 72% of the time. While this is significantly better than chance, there is definitely room for improvement.

As an aside, unexpectedly, we see that removing the stop words, the step which perhaps makes the most sense from a purely linguistic point of view, produced the worst model, and the combined process, which includes removing the stop words, is the second worst. The differences are small enough that perhaps it is statistical noise, but this was not what we would have expected. Perhaps Frederick Jelinek, the noted computational linguist and scholar, was right when he said of his tests “Every time I fire a linguist, the performance of the speech recognizer goes up. (1)”

As an aside, one set of data which we do not have access to, which would be interesting for comparison, is an evaluation of how well humans would do with this task. Judging sarcasm, especially with just text in the absence of tone and gesture, might be difficult for human speakers as well. Would they be noticeably better than 70 to 72 percent? Or maybe worse? We wish we knew so we could compare and judge our algorithm against that standard, given the perceived difficulty of the task.

ADDITIONAL CONSIDERATIONS:

Unfortunately, word2vec only allowed for a generic “binomial family” algorithm. It would have been nice to test different algorithms here: Decision Trees, or Naive Bayes. Time did not allow, but this would be a potentially fruitful avenue of investigation.

Similarly, we could have tested other items, such as number of upvotes or downvotes, or perhaps the username of the author, to determine if any of these may have had a useful input in determining if a comment is sarcastic.

APPENDIX:

SOURCE DATA:
Data provided by Mikhail Khodak, Nikunj Saunshi and Kiran Vodrahalli: "A Large Self-Annotated Corpus for Sarcasm"

https://www.kaggle.com/sherinclaudia/sarcastic-comments-on-reddit

CITATIONS:
(1) Hirschberg, Julia (July 29, 1998). 'Every time I fire a linguist, my performance goes up', and other myths of the statistical natural language processing revolution (Speech). 15th National Conference on Artificial Intelligence, Madison, Wisconsin. Invited speech.

CODE AND OUTPUT:
See attached Red file for code, and see attached HTML file for output showing selected graphs and data.