Assignment 1: Sarcasm Classification (100 points)

CS 410/510 Natural Language Processing Fall 2022
Due Thursday, 10/13/2022, 11:59pm

This is an individual assignment. Although you may discuss the questions with other students, you should not discuss the answers with other students. All code and written answers must be entirely your own.

1 Description

The main goal of Assignment 1 is to build a classifier that will predict whether a piece of text is "sarcastic" or "not sarcastic/regular". Your task is to see how accurate a classifier you can build, depending on the machine learning approach and on the various features you will be asked to implement.

1.1 Data

The data[1][1] is a set of news headlines collected from two news website: The Onion[2], which aims at producing sarcastic versions of current events, and HuffPost[3], which provides the set of real (and non-sarcastic) news headlines.

Warning: *As is the case with most 'natural language' text, this data was collected from public websites and is mostly unfiltered. Therefore, it is possible that some text may be disturbing or you may not agree with it.*

The data file provided to you is in json format and is formatted with one instance/headline per line. Each instance consists of three attributes:

- is_sarcastic: 1 if the headline is sarcastic, otherwise 0
- headline: the headline of the news article
- article_link: link to the original news article (this is just for your reference, you will not be using this information in this assignment).

Reading the data: In Python, data can be read using the following function:

```
def parseJson(fname):
    for line in open(fname, 'r'):
        yield eval(line)
```

Example usecase:

```
data = list(parseJson('./Sarcasm_Headlines_Dataset.json'))
```

[1] https://github.com/rishabhmisra/News-Headlines-Dataset-For-Sarcasm-Detection
[2] https://theonion.com/
[3] https://www.huffingtonpost.com/

1.2 Programming component

You may attempt to write all code yourself. If you wish, you may use publicly available libraries and pieces of code as long as you credit the source(s).

Feature engineering

An important part of text classification is choosing the set of features and their representation that would help you in building the most accurate classifier. Some of the features that you have already seen in class include n-grams. As part of this assignment you will further experiment with a variety of features to find the best performing features for your models. As an example, you can use any of the feature types listed below or use additional features you come up with yourself (NOTE: each of these counts as ONE feature type, later described in 1.2.2):

- Num words: the number of words per headline
- N-grams: unigrams, bigrams, and trigrams
- Cue words: the initial unigram, bigram, and trigram in a post
- Repeated punctuation: features to capture punctuation such as ??, ??????, !!!, or ?!.
- Part-of-speech tags: features to count the number of nouns, verbs and adjectives in the text (count_noun, count_verb, and count_adj)
- NRC Emotion features [2]: we also include the NRC Emotion lexicon, which is a collection of unigrams and their associated emotion and emotion intensity scores. You may use this lexicon to derive a combination of features such as:

  emo words: the number of [emo] words in the headline, where [emo] is one of the eight emotions - anger, anticipation, disgust, fear, joy, sadness, surprise or trust. If you use all eight emotions, you will have eight different features, such as anger_words, disgust_words, and so on.

  emo intensity: the average intensity of each emotion. As an example, similarly to the previous feature, if you choose to use all eight different emotions, you will obtain eight features. For instance, for a headline, anger_intensity = sum of intensities of all 'anger' words / total number of 'anger' words in the headline.

Classification models

You will be using the `scikit-learn` library to do the homework. See the `scikit-learn` documentation for many useful functions available. You are to experiment with two classification models: Naive Bayes and Logistic Regression.

Input: features you have selected.

Output: a label $y \in 0, 1$, indicating whether the headline is "sarcastic" (1) or "regular" (0).

In this assignment you will build your classifiers using features as specified below:

1. N-grams: this model should use a combination of unigrams, bigrams and trigrams as features. The combination you use is up to you and you should experiment to find the one that performs best.

2. Other features: this model can use any combination of features (see section 1.2). You must *experiment with at least 3 feature types other than n-grams* for this model, and discuss whether they were beneficial or harmful to your model (remember, not all features add positive value).

Evaluation

You will use 10-fold cross-validation to evaluate your models. In addition to creating train and test splits, it is generally advisable to also create a validation/development set which you can use to prevent overfitting of your model. As evaluation metric, please provide the accuracy and F-score.

1.3 Written component

1. Present and discuss the results of the two classification models and different feature sets.
2. Describe your best model and a justification of the features that you used or did not use. Why do you think they are helpful or not? You must explicitly list the features you tried here.
3. Conduct an error analysis. For your best performing model, select 3 interesting examples that did not work. Indicate why you think they were classified incorrectly.
4. If you could add additional features, what features do you think would help improve performance?

2 Grading

Please upload to Canvas the link to your Jupyter Notebook **hw1.ipynb** containing the solutions to the programming and written components. Please clearly indicate your name in the notebook.

- Programming component (60 points total):
  - Your program should read the headlines json file (please do not modify the file name), create the relevant features, run 10-fold cross-validation, and print the model's average accuracy and F-score. (40 points)
  - Code documentation: your jupyter notebook should be well documented including associated descriptions for each code text as well as all citations for any source used during this assignment. (10 points)
  - Model performance: your system will receive up to 10 points depending on whether the models produced F-score above or below our baseline thresholds.
- Written report (40 points): In the jupyter notebook, please include the answers to the four questions, each worth 10 points.

References

[1] Rishabh Misra and Prahal Arora. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*, 2019.

[2] Saif Mohammad and Peter Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34, 2010.