CS 585 – Fall 2023 – Homework 2 (100 total points)

Due Wednesday September 27, 11:59pm

GOALS

- To perform a text classification experiment and examine the results
- To become familiar the scikit-learn machine learning library
- Optional exercise: To gain hands-on experience with word embeddings

DATA

For this homework you will build a text classification model to identify **clickbait**, or text headlines whose main purpose is to attract attention and entice readers to follow a link.

You will use data that has been created and shared by other NLP researchers on GitHub:

- Positive examples: https://github.com/pfrcks/clickbait-detection/blob/master/clickbait
- Negative examples: https://github.com/pfrcks/clickbait-detection/blob/master/not-clickbait

You do not need to read any research papers on clickbait detection to complete this homework, but you may enjoy reading the original publication that shared this dataset.

TOOLS

In this homework you are asked to use some functions from scikit-learn , an open-source python library widely used by data scientists.

You should use Python to complete this homework assignment. Other programming languages will not be accepted.

NOTE ON MODEL PERFORMANCE

The goal of is homework is to execute an NLP experiment and present the results clearly to others (in this case, the class TAs) can read clearly. Different students will see different results, if you see lower metrics than others be assured that this will not lower your grade.

WHAT TO SUBMIT

Please upload or submit the following in Blackboard:

- For Problems 1 and 3-7, please upload to Blackboard:
  - One Jupyter notebook (.ipynb file) with cell output, showing your work for both datasets.
  - A PDF copy of the exact same notebook (same code and same output)
- For Problems 2 and 8 (no-code Q&A problems): Enter your written answers in Blackboard with your HW submission.

PROBLEM 1 – Reading the data (5 pts)

- Using Python, read in the 2 clickbait datasets (See section DATA), and combine both into a single, shuffled dataset. (One function to shuffle data is numpy.random.shuffle)
- Next, split your dataset into **train, test, and validation** datasets. Use a split of 72% train, 8% validation, and 20% test. (Which is equivalent to a 20% test set, and the remainer split 90%/10% for train and validation).
    - If you prefer, you may save each split as an index (list of row numbers) rather than creating 3 separate datasets.
- What is the "target rate" of each of these three datasets? That is, what % of the test dataset is labeled as clickbait? Show your result in your notebook.

PROBLEM 2 – Baseline Performance (10 pts – Answer in Blackboard)

- Assume you have a trivial baseline classifier that flags **every** text presented to it as clickbait. What is the precision, recall, and F1-score of such a classifier on your test set? Do you think there is another good baseline classifier that would give you higher F-1 score?

PROBLEM 3 – Training a single Bag-of-Words (BOW) Text Classifier (20 pts)

- Using scikit-learn pipelines module, create a Pipeline to train a BOW naïve bayes model. We suggest the classes CountVectorizer and MultinomialNB. Include both unigrams and bigrams in your model in your vectorizer vocabulary (see parameter: **ngram_range**)
- Fit your classifier on your training set
- Compute the precision, recall, and F1-score on both your training and validation datasets using functions in sklearn.metrics. Show results in your notebook. Use "clickbait" is your target class (I.e., y=1 for clickbait and y=0 for non-clickbait)

ALTERNATIVE: If you are already well-versed in Naïve Bayes, you may select another bag-of-words classifier for this problem. Your method should have some way to select top features or key indicators, mapped to words or n-grams in your vocabulary, so that you can complete the remaining problems

PROBLEM 4 – Hyperparameter Tuning (20 pts)

Using the ParameterGrid class, run a small grid search where you vary at least 3 parameters of your model

- **max_df** for your count vectorizer (threshold to filter document frequency)
- **alpha** or smoothing of your NaïveBayes model
- One other parameter of your choice. This can be non-numeric; for example, you can consider a model with and without bigrams (see parameter "ngram" in class CountVectorizer)

Show metrics on your **validation** set for precision, recall, and F1-score. If your grid search is very large (>50 rows) you may limit output to the highest and lowest results.

ALTERNATIVE – If you used a method other than Naïve Bayes in Problem 3, then be sure it is clear what metrics you tuned in Problem 4.

PROBLEM 5 – Model selection (10pts)

Using these validation-set metrics from the previous problem, choose one model as your **selected model**. It is up to you how to choose this model; one approach is to choose the model that shows the highest F1-score on your training set.

Next apply your selected model to your test set and compute precision, recall and F1. Show results in your notebook

PROBLEM 6 – Key Indicators (10pts)

Using the log-probabilities of the model you selected in the previous problem, select **5 words** that are strong **Clickbait indicators**. That is, if you needed to filter headlines based on a single word, without a machine learning model, then these words would be good options. Show this list of keywords in your notebook.

You can choose how to handle bigrams (e.g., "win<space>big"); you may choose to ignore them and only select unigram vocabulary words as key indicators.

PROBLEM 7 – Regular expressions (10pts)

Your IT department has reached out to you because they heard you can help them find clickbait. They are interested in your machine learning model, but they need a solution today.

- Write a regular expression that checks if any of the keywords from the previous problem are found in the text. You should write one regular expression that detects any of your top 5 keywords. Your regular expression should be aware of word boundaries in some way. That is, the keyword "win" should not be detected in the text "Gas prices up in winter months".
- Using the python re library – apply your function to your test set. (See function re.search). What is the precision and recall of this classifier? Show your results in your notebook

PROBLEM 8 – Comparing results (15pts – Answer in Blackboard)

- Compare your rules-based classifier from the previous problem with your machine-learning solution. Which classifier showed the best model metrics? Why do you think it performed the best? How did both compare to your trivial baseline (Problem 2)?
- If you had more time to try to improve this clickbait detection solution, what would you explore? (There is no single correct answer to this question; review your results and come up with your own ideas)

OPTIONAL LEARNING EXERCISE – Applying word embeddings (Not graded)

[This is an optional exercise and does not need to be included in your homework submission]

Your IT department loves your solution and wants you to add more keywords that they can search for. Your teammate then suggests using word embeddings to create a larger set of keywords.

- Using some word embedding dataset, find 5-10 words that are similar to your top-5 keywords from Problem 6. It is up to you to decide how to select the most "similar" words.
- Create a classifier that uses this new set of words. This can be as simple as flagging a text as clickbait if any of these words appear.
- What is the precision, recall, and f-score of this new word embedding based classifier? Did you improve on either your machine learning or rules-based (regular expression-based) classifier?

A note on Word2Vec - You can use this code snippet to find similar words; note the binary file "`GoogleNews-vectors-negative300.bin`" needs to be downloaded in advance.

```
import gensim

word2vec = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-
negative300.bin', binary=True, limit=100000)

word2vec.most_similar("blue")
```

```
[('red', 0.7225173115730286),
 ('purple', 0.7134225964546204),
 ('white', 0.6606027483940125),
 ('maroon', 0.6557417511940002),
 ('colored', 0.6422423720359802),
 ('orange', 0.6421891450881958),
 ('yellow', 0.6376120448112488),
 ('teal', 0.6356961131095886),
 ('pink', 0.6343377232551575),
 ('pale_blue', 0.6308072209358215)]
                                                                                [146]:
```

REFERENCES

Agrawal, A. (2016). Clickbait detection using deep learning. In 2nd international conference on next generation computing technologies (NGCT) (pp. 268–272).