Homework 2: Text classification (CS 2731 Fall 2023)

Due 2023-10-05, 11:59pm. *Instructions last updated 2023-09-26.*

Part 1: Learning weights in logistic regression

You are training a classifier for reviews of a new product recently released by a company. You design a couple of features, x_1 and x_2. You will be using logistic regression. With an initialization of the weights w_1, w_2 and b (the bias) all set = 0 and a learning rate $\eta = 0.2$, calculate the weights after processing each of the following 3 inputs:

1. $$x_1 = 2, x_2 = 1, y = 1$$
2. $$x_1 = 1, x_2 = 3, y = 0$$
3. $$x_1 = 0, x_2 = 4, y = 0$$

During calculations, keep at least 3 significant digits for values. Points will not be taken off for slight differences due to rounding.

Deliverables for Part 1

- In the report, show your work in calculating the values of the weights after training on each data point. Briefly comment on any shift in weights from positive to negative or negative to positive and why this was the case.

Part 2: Implement a politeness classifier

In this portion, you will design and implement a program to classify if an online comment is polite or not. You can use any packages you want for this (scikit-learn, spaCy, NLTK, Gensim, code from Homework 1, etc), but these must be specified in the README.txt file, along with version numbers for Python and all packages. We will attempt to run your code on a held-out test set, so the exact environment must be specified. If you will be using a language other than Python, please let us know before submitting. Your script should be able to take the name of a dataset as a single keyword argument.

Dataset

Here is the dataset that you should download for this assignment:

- `politeness_data.csv`. This dataset has 3 fields:
  - `id`: a unique identifier for the post
  - `text`: the text of the comment
  - `polite`: a binary class label of whether or not the comment was annotated to be polite (1) or not (0) This dataset consists of requests among Wikipedia editors posted on user talk pages, as well as posts on the coding help forum Stack Exchange (see the Stanford Politeness Corpus, Danescu-Niculescu-Mizil et al. 2013).

2.1 Feature-based logistic regression models

In this section, you will build a logistic regression model based on bag-of-word features and/or features of your own design. You can do whatever preprocessing you see fit. You will report performance using 5-fold cross-validation on the dataset, which you will set up.

Tasks for section 2.1

Implement and try the following feature and model combinations:

- *Logistic regression with bag-of-words (unigram) features*. Build a logistic regression classifier that uses bag-of-words (unigram) features.
- *Logistic regression with your own features/change in preprocessing*. Design and test at least two of your own custom features or change the preprocessing in at least one way. Note that these features can be used in conjunction with bag-of-words features or by themselves. Possible features/changes to add and test include:
  - Tf-idf transformed bag-of-words features
  - Changing count bag-of-words features to binary 0 or 1 for the presence of unigrams
  - N-gram features (sequences of words) beyond the single words used for the bag-of-words features
  - Different preprocessing (stemming, different tokenizations, stopword removal)
  - Reducing noisy features with feature selection
  - Counts from custom word lists
  - Static word embeddings of your choice
  - Any other custom-designed feature (such as length of input, number of capitalized words, etc)

In the report, please provide:

- A table of performance scores for models trained on each set of features. Include accuracy as well as precision, recall, and f1-score for the positive (polite) class.
- For each feature or change in input text processing:
  - Describe your motivation for including the feature
  - Discussion of results: Did it improve performance or not? (Either result is fine. It is not necessary to beat logistic regression with unigram features.)
- For a feature-based model of your choice:
  - List the top 2 most informative features that are mostly strongly positively and negatively associated with politeness. Discuss if you find these surprising and any other comments you might have. You may adapt code provided by the instructor in the Naive Bayes example (notebook here), use another source online, or write your own.
  - Do an error analysis. Provide a confusion matrix and sample multiple examples from both false negatives and false positives. Do you see any patterns in these errors? How might these errors be addressed with different features or if the system could understand something else? (You don't have to implement these, just speculate.)

2.2 Static word embeddings with feedforward neural network

In this section, you will build and evaluate a feedforward neural network that uses pre-trained static word embeddings as input. To represent the document, you can take the average word embeddings of the input sentence or choose another function. You can choose which activation function to use and other hyperparameters. You will again use 5-fold cross validation on the dataset. There is no need for this model to outperform the logistic regression model you made.

Tasks for section 2.2

- Implement a feedforward neural network with static word embeddings as input.

In the report, please provide:

- Performance scores for this model. Include accuracy as well as precision, recall, and f1-score for the positive (polite) class. This can be an additional row in the table with other performance scores.
- Discuss the motivation for any choices you made as far as word embedding types, pretraining dataset, and/or how you represented the document, or if you experimented with multiple of these options.
- Discuss the motivation for any choices you made as far as network architecture (number and dimensions of hidden layers) or hyperparameters (learning rate, number of epochs, etc). Note if you experimented with any of these options.

Notes

- 5 bonus points will be awarded for the best-performing logistic regression classifier and 5 bonus points for the best neural network classifier, as measured by accuracy on our held-out test set.
- Don't feel like you need to write things from scratch; use as many packages as you want. Google and Stack Overflow and NLP/ML software documentation are your friend! Adapting and consulting other approaches is fine and should be noted in comments in the code and/or in the `README.txt`. Just don't use complete, fully-formed implementations for this (including from generative AI tools). Use resources as an aid, not as a final product.
- Optionally, you may incorporate any form of regularization that you like
- This homework is designed to be able to be run on a laptop with CPUs, not GPUs. Let the instructor/TA know if you are having difficulty completing it with the resources you have.

Deliverables

- A README.txt file explaining
  - how to run your code
  - the computing environment you used; what programming language you used and the major and minor version of that language; what packages did you use in case we replicate your experiments (a `requirements.txt` file for setting up the environment may be useful if there are many packages).
  - any additional resources, references, or web pages you've consulted
  - any person with whom you've discussed the assignment and describe the nature of your discussions
  - any generative AI tool used, and how it was used
  - any unresolved issues or problems
- Your report with results and answers to questions in Part 1 and Part 2, named `report_{your pitt email id}.pdf`.
- Your code in a file named `hw2_{your pitt email id}_test.py`. **This script should be able to take the name of a new dataset, which will be in the same format as the training set, as a single keyword argument, as in the command `python hw2_{your pitt email id}_test.py data.csv`.** This script can either load your trained model (which also needs to be submitted) or train in a reasonable amount of time with the `politeness_data.csv` assumed to be in the current working directory. We will attempt to run this code on a held-out test set.
- Any additional files needed to run the code. If you draw your pretrained static embeddings from a file that you downloaded, upload it to Canvas if it's <400MB. Otherwise provide a direct URL link to the embeddings and the name and version.

Please submit all of this material on Canvas. We will grade your report and attempt to run your code.

Acknowledgments

This assignment is inspired from a homework assignment by Prof. Diane Litman. Data is from Danescu-Niculescu-Mizil et al. 2013.