CS 590 NLP

HW2

Language Models, Naïve Bayes, Logistic Regression

Due 10/21 11:59 pm

Overall Goal:

In this homework you will work with LMs to analyze a dataset and Logistic Regression to classify the same dataset.

Useful python packages:
For the LM, you may find the nltk package useful: https://www.nltk.org/api/nltk.lm.html Or you can implement the LM manually if you prefer. (Following the n-gram model discussed in class.)
For the Logistic Regression, there is also a nltk package: https://www.nltk.org/_modules/nltk/classify/scikitlearn.html Or there is a sklearn package: https://pythonprogramming.net/sklearn-scikit-learn-nltk-tutorial/ You may also implement this manually, but this would require also implementing the loss algorithm and stochastic gradient descent for training, so I would recommend against this.

Dataset
The dataset that will be used is the OLID dataset. This consists of offensive tweets, so be cautioned when viewing the dataset. You can find the training and test sets here: https://www.kaggle.com/datasets/feyzazkefe/olid-dataset Note for the LM and LR tasks you will only need to worry about the first label (Offensive or Non-offensive) (Task A). The training set (**olid-training-v1.0.tsv)** will be used for training the LM and LR, while the test set (**testset-levela.tsv**) and labels **(labels-levela.csv)** will be used for testing and analyzing your models. Take time to understand how the training set and test set are laid out.

Language Model Tasks

You will create 2 required functions for LMs (you may create more functions for your own use, but you need to at least create these two as specified):

train_LM(path_to_train_file)
This method trains up to a bigram LM and returns the trained LM. **The format for the train file should follow the same format as the olid-training data file!** The LM links have useful steps to help you train a LM in nltk.

test_LM(path_to_test_file, LM_model)
This method tests an LM on a test file. That is, for each instance in the test file, it assigns a MLE score (refer to in class notes for how we approach this) for each test text. The function then generates an output file as the same format as the test file but adds a new column with the MLE score. **The format for the input file should follow that of the olid test file.**

Once these functions are implemented use them to accomplish the following tasks.

1. Create 3 LMs:
    a. LM_full -> trained on all the olid training data.
    b. LM_not -> trained on all the **non-offensive (NOT)** training data
    c. LM_off -> trained on all the **offensive (OFF)** training data
2. For each LM, test on: 1) the full test set, 2) the not offensive subset, 3) the offensive subset
3. For each test file, make observations (e.g. averages of scores for LM_full on the offensive subset compared to LM_not) and analyze if your LMs seem to be successfully capturing the language they were trained on.
4. Write out the observations of LMs to a report document to hand in alongside your code. I will be looking for multiple comparisons of LMs across test sets. Additionally, any problems encountered or errors seen during creation of the LM/testing will help demonstrate your thoughtfulness in this homework.

Logistic Regression Model Tasks

You will create 2 required functions for LRs (you may create more functions for your own use, but you need to at least create these two as specified):

train_LR_model(path_to_train_file)
This method trains a logistic regression model on some set of features and returns that trained model. These features are up to you. You may leverage the text vectorizer/tfidf available in nltk or you may define a set of features by observing the text. (e.g. number of words in an offensive word lexicon, length of text, amount of punctuation, etc.) **The format for the train file should follow the same format as the olid-training data file!**

test_LR_model(path_to_test_file, LR_model)

This method tests a trained LR model on some test file and outputs a test file in the same format as the input test file but with 2 columns added: 1) probability of that text being offensive, 2) class prediction (OFF, NOT). **The format for the input file should follow that of the olid test file.**

Once these functions are implemented use them to accomplish the following tasks.

1. Train a LR model on the entire train set.
2. Test the trained model on the test set and produce predictions for all test texts.
3. Analyze your accuracy of your LR model and try to improve it. This may be done by: adding other features in, improving your preprocessing of the text, changing the hyperparameters of the LR model for training, etc.
4. Write the analysis and attempts to improve the accuracy as observations in a report document to be handed in alongside code. Note that even if changing something causes the accuracy to drop, this should be reported along with your reasoning of the drop/improvement in accuracy. This report is how you can demonstrate that you are thinking through the problem and give you practice for your final project.

Additional Tips/Guidance

1. The accuracy you achieve will not determine your final grade for this homework. Rather your thoughtfulness in approaching and analyzing your models. This means that lack in analysis will receive lower scores as indicated in the grading scale.
2. Note that preprocessing the text is up to you. This assignment is not specifically evaluating preprocessing like the first assignment, but here you have a chance to practice any preprocessing for your final project and future assignments.
3. Get familiar with the NLTK libraries on your own on small sets of data. Just copying and pasting the code without fully understanding it will end up being detrimental as you may not be able to accurately give proper analysis and observations of results (especially for the LR model where you are expected to make improvements/changes).
    a. Do not just copy and past the NLTK class code. You may use portions from the nltk classes, but too much unused code from classes will result in lost points due to poorly written code. Experiment with the code and only import what you need.
4. DO NOT SHARE CODE. I have linked NLTK libraries which will be useful for this assignment. You may also come to me to discuss/figure things out.
5. Start the assignment early. You will have 3 weeks to complete this assignment, which is plenty of time if you start early any familiarize yourself with the libraries. **IF YOU WAIT UNTIL THE LAST MINUTE TO START, YOU WILL BE LESS LIKELY TO DO WELL.** I won't be granting any additional extensions so each late assignment with follow the late grade policy.
6. Ask questions/approach like a researcher. Think like this is your chance to explore NLP models and analyze their effectiveness.

Grading

Assignment will be graded as follows:

Description	**Points**
Code Runs	10
LM Implementation/Tasks	25
LR Implementation/Tasks	25
Report analysis and observations	25
Documentation (Comments, functions, etc)	15
Total:	**100**
Extra Credit (NB tasks)	**20**

EXTRA CREDIT (20 pts): Naïve Bayes Model Tasks

For extra credit, you may implement and test the Naïve Bayes model (in NLTK) similarly to how the LR is being investigated. Note however, you won't be using features but the training test itself.
(https://www.nltk.org/_modules/nltk/classify/naivebayes.html)
(https://pythonprogramming.net/sklearn-scikit-learn-nltk-tutorial/)

NOTE: Extra credit will not be counted for any late submissions!

You will create 2 required functions for NBs (you may create more functions for your own use, but you need to at least create these two as specified):

train_NB_model(path_to_train_file)
This method trains a naïve bayes model on the training text and returns that trained model. **The format for the train file should follow the same format as the olid-training data file!**

test_NB_model(path_to_test_file, NB_model)
This method tests a trained NB model on some test file and outputs a test file in the same format as the input test file but with 2 columns added: 1) probability of that text being offensive, 2) class prediction (OFF, NOT). **The format for the input file should follow that of the olid test file.**

Once these functions are implemented use them to accomplish the following tasks.

1. Train a NB model on the entire train set.
2. Test the trained model on the test set and produce predictions for all test texts.
3. Analyze your accuracy of your NB model and compare to the LR model. Compare and contrast how the models did on different labeled data (ie NOT vs OFF texts).
4. Write the analysis and comparisons to your report file. This report is how you can demonstrate that you are thinking through the problem and give you practice for your final project.