CS 491/691 Project 2

Natural Language Processing

Please submit on zip file to Webcampus in a folder named Project2. You are provided with test_script.py. I will use the test_script.py to evaluate your implementations. Make sure you can run test_script.py without errors. Assume you are running test_script.py in Project2 with helpers.py, hmm.py, naive_bayes.py and logistic_regression.py. Submit a zip file with all the applicable py files as well as your README.txt.

Helper Functions (10 points)

Please implement the following functions in a file titled helpers.py.

`def load_pos_data(fname):`
This function should load the POS tagging data from the file fname and return the stored data in whatever format you'd like. The returned data will be passed as an argument in your HMM code. Since there will be a very large vocabulary, you can remove the least common words to make it more manageable. You can do any kind of preprocessing you like, actually. Just make a note of it in your README.txt.

`def load_spam_data(dirname):`
This function should load the spam/ham data from the directory dirname (dirname should have two subdirectories with spam and ham data) and return the stored data in whatever format you'd like. You should convert each spam/ham file into a Bag of Words vector. Since the vocabulary will be VERY LARGE, you can remove common and uncommon words and do other kinds of preprocessing you see fit. Just make a note of it in your README.txt The only requirement is that the function should have two return values (train_data and test_data). Use 80% of the data for training and 20% for testing. The returned data will be passed as an argument in your Naïve Bayes and Logistic Regression code.

Hidden Markov Models for POS Tagging (30 points)

Please implement the following functions in a file titled hmm.py.

`def train_hmm(train_data):`
This function should take your train_data from helpers.load_pos_data(fname) and generate transition and emission tables. Both of these should be returned from your function. You can store them however you'd like!

`def test_hmm(test_data, hmm_transition, hmm_emission):`

This function should take your test_data from helpers.load_pos_data(fname), transition table and your emission table. For each item in test_data (which can have more than one sample), you should compute the highest probability sequence of tags (using the Viterbi algorithm). Compare the predicted tags with the tag labels in test_data and return an average accuracy as well as a per-sequence accuracy.

Naïve Bayes (30 points)

Please implement the following functions in a file titled naive_bayes.py.

`def train_naive_bayes(train_data):`
This function should calculate class probabilities as well as conditional word probabilities for each class (aka, train a Naïve Bayes model). You can store this model however you'd like, and it should be the only thing returned from the function.

`def test_naive_bayes(test_data, model):`
This function should take your take your test data and model as input. For every sample in test_data, use the model to predict the most likely class, compare this class with the label and calculate overall accuracy. The overall accuracy should be the only thing returned from this function.

Logistic Regression (30 points)

Please implement the following functions in a file titled logistic_regression.py.

`def train_logistic_regression(train_data):`
This function should take the training data and train a logistic regression model. It should return the trained model (in whatever format you'd like to store it).

`def test_logistic_regression(test_data, model):`
This function should take your take your test data and model as input. For every sample in test_data, use the model to predict the most likely class, compare this class with the label and calculate overall accuracy. The overall accuracy should be the only thing returned from this function.