Problem 1:
a) To train: problem-1.py train
b) To tune: problem-1.py tune
c) To test: problem-1.py test

Best accuracy on training(92.9745889387145%): number_of_epocs, size_of_minibatch, learning_rate = 1,1,1

Best accuracy on devel(96.56630365150545%): number_of_epocs, size_of_minibatch, learning_rate = 7,3,0.1
Other trials(92.60089686098655%): number_of_epocs, size_of_minibatch, learning_rate = 2,3,0.1
other trials(95.17488789237667%): number_of_epocs, size_of_minibatch, learning_rate = 5,4,0.1

Best accuracy on test(96.58886894075404%): number_of_epocs, size_of_minibatch, learning_rate = 7,3,0.1
Using the saved model and saved hyperparameters for testing

Problem 2:
Using both unigram and bigram features
a) To train: problem-2.py train
b) To tune: problem-2.py tune
c) To test: problem-2.py test

Best accuracy on training(93.27354260089686%): number_of_epocs, size_of_minibatch, learning_rate = 1,1,1

Best accuracy on devel(97.8603459320948%): number_of_epocs, size_of_minibatch, learning_rate = 7,3,0.1
Other trials(95.42600896860986%): number_of_epocs, size_of_minibatch, learning_rate = 4,5,0.1
other trials(97.4439461883408%): number_of_epocs, size_of_minibatch, learning_rate = 8,5,0.1

Best accuracy on test(95.69120287253142%): number_of_epocs, size_of_minibatch, learning_rate = 7,3,0.1
Using the saved model and saved hyperparameters for testing

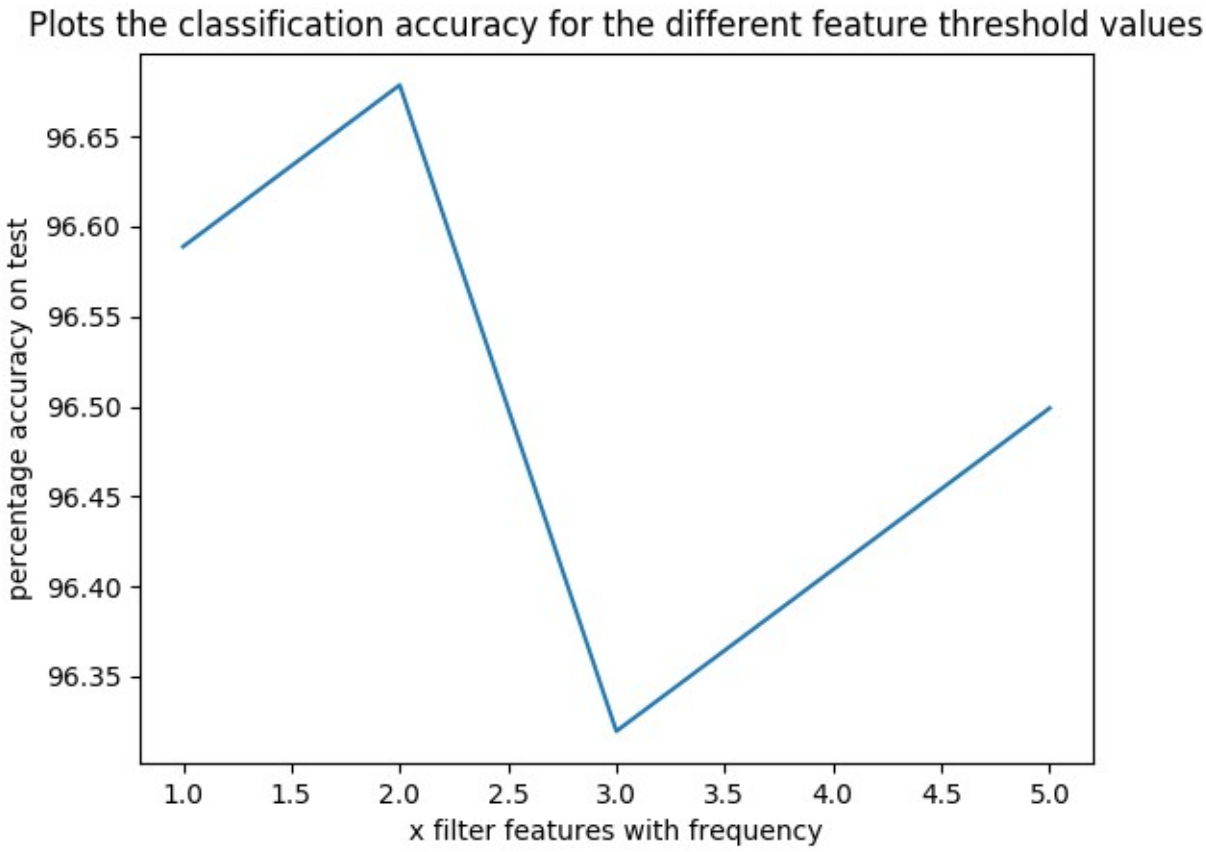
Problem 3:
Filter features with frequency 1:
% = 97.45035233824471 accuracy on train
96.588848833035 on test

Filter features with frequency 1 and 2:
98.33013025838137 accuracy on train
% = 96.67863554757629 on test

Filter features with frequency 1 and 2 and 3:
97.15566944266496 accuracy on train
% = 96.31956912028726 on test

Filter features with frequency 1 and 2 and 3 and 4:
97.014734144779 accuracy on train
% = 96.40933572710951 on test

Filter features with frequency 1 and 2 and 3 and 4 and 5:
96.79692504804612 accuracy on train
% = 96.49910233393177 on test



The classification acuracy doesnt change much with filtering. In fact with very little features itself the classification accuracy is good i.e 96.67863554757629 on test data when we remove features of frequency 1 and 2.
The filering functionality is included in run_LR_filter() method(problem-2.py) line 250.

Problem 4:

The top 20features of ham class: to, you, I, a, the, and, is, in, u, i, me, for, my, of, have your, on, to, it,that.
The top 20features of spam class: are, out, 4, this,from, want, only, by, Call, text, who, &, FREE,phone, reply, txt, new, will_be, claim, mobile

The algorithm is overfitting since some words which can be spam/ham class are found in both. The classifier here tries to learn all the words rather than most important or key words. Hence adding a stop words list can be a good option.

Problem 5:

http://www.cs.cmu.edu/~wcohen/postscript/speech-acts.pdf - Learning to Classify Email into "Speech Acts"

The task here is to classify the email based on the intent of the sender. If two people are working together and discussing the schedule, the classification algorithm can mark it as to-do, request, commitment, remainders,etc.
It uses the ontology of words, noun and verb to identify and calssify the email using machine learning

techniques. It also analyses the text in the users inbox to regularize and in a sense linguistic behaviour of words is condidered rather than the space of speech acts.
The experimental results show that its better than the baseline SVM results. Baseline F score is 25 and their approach is 45 and above. They have also explored bigrams, POS tags and person phrases.
One limitation is that the ontology has to be given to such a classifier and cannot be automatically constructed!
The context of the message and the structure of the language hasnt been explored and left for future work.