This project will introduce you to natural language processing (NLP) and machine learning using support vector machines (SVMs). Your task will be to train and use an SVM that classifies word strings as sentences or non-sentences.

There are 5 files in this project:
- Classifier.py
- lex/True_Train
- lex/False_Train
- lex/True_Test
- lex/False_Test

Classifier.py is where you will be coding your solutions to the project. The four files in lex contain sentences that you will be using to train and test your SVM classifier. The files beginning with 'True' are the positive examples and those beginning with 'False' are the negative examples. Each sentence is 10 tokens long, where a token can be punctuation or a word.

You will need to install NumPy, the Python Natural Language Toolkit (NLTK), the Brown Corpus for NLTK, and the Python SKLearn packages:

```
sudo pip3 install numpy
sudo pip3 install scipy
sudo pip3 install nltk
sudo pip3 install -U scikit-learn (for more information: http://scikit-learn.org/stable/install.html)
```

To download the Brown Corpus, see [http://www.nltk.org/data.html](http://www.nltk.org/data.html) and follow the instructions (you may either download everything or look specifically for the corpus called 'brown')

You will be using excerpts from the Brown Corpus included within NLTK to calculate the probabilities of bigrams (or sequential left-to-right word pairs). You are encouraged to use NLTK functions where possible but should not import any additional modules. More information on bigrams in NLTK can be found at [http://www.nltk.org/_modules/nltk/util.html](http://www.nltk.org/_modules/nltk/util.html). More information about calculating probabilities in NLTK can be found at [http://www.nltk.org/_modules/nltk/probability.html](http://www.nltk.org/_modules/nltk/probability.html) (hint: look at the NLTK frequency distribution class). With these probabilities, you will first create a 9-dimensional feature vector for each sentence in True_Train and False_Train. If your sentence is “Tired from her long day, Susan slept soundly.”, then the bigrams for this sentence are [('Tired', 'from'), ('from', 'her'), ('her', 'long'), ('long', 'day'), ('day', ','), (',' , 'Susan'), ('Susan', 'slept'), ('slept', 'soundly'), ('soundly', '.')].

**Being sure to preserve this order,** the 9-dimensional feature vector would correspond to the probabilities of those bigrams occurring in the Brown Corpus, ignoring differences in capitalization ( ('Tired', 'from') should be treated as the same bigram as ('tired', 'from')). These feature vectors will be the input to your SVM. You will then train an SVM on these positive and negative examples and evaluate it's performance on the sentences in True_Test and False_Test. More information on creating and running your SVM can be found at [http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) (hint: look at the first example of this page for usage information). To test your SVM, you will similarly create 9-dimensional feature vectors for each test word-string and use your SVM to predict whether each test word-string is a sentence or non-
You can train the SVM on the sentences in True_Test and False_Test but the F1 scores you receive may be inflated due to overfitting. You may also optionally train your SVM using additional examples of sentences and non-sentences to try and improve your score (a large part of your grade will come from your SVM's performance on a separate test set of sentences and non-sentences similar to those in the test sets you were provided). Def svmpredict(self) will print your SVM's F1 score on the test set of sentences (F1 scores are given between 0-1 and a score closer to 1 signifies better SVM classification; see http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html for more information). As a reference, you should be getting an F1 score of at least 0.6 on this dataset. If you are getting lower values, experiment with adding multipliers to the values in your feature vector and changing your SVM parameters.

You will be implementing what is described in the previous paragraph in the following 4 methods. You should ONLY modify the following 4 methods and MAY NOT change their arguments or their return statements:
- def get_standard_features(self, sents):
  - Find bigram probabilities for each string in sents and place in features array
- def compute_corpus_bigrams(self):
  - Compute the frequency distribution of bigrams in the excerpted Brown corpus
- def runsvm(self):
  - Train an SVM from the feature vectors of strings in the training set
- def svmpredict(self):
  - Use your SVM to predict classifier scores for a set of test strings

You may define additional class members in the above 4 methods but should NOT do so anywhere else.

**BONUS**

Bonus credit will be given for successfully implementing two additional functions:
- def get_extended_features(self, sents):
  - Find bigram and trigram probabilities for each string in sents and place in features array
- def compute_corpus_trigrams(self):
  - Compute the frequency distribution of trigrams in the excerpted Brown corpus

In these functions, you will be extending the 9-dimensional feature vector to a 17-dimensional feature vector that also includes trigram probabilities. If you choose to implement these functions, you can change def main(ftype='standard') to def main(ftype='extended') so that you can test your code.

**SUBMISSION INSTRUCTIONS AND GRADING**

Your submission must run under Python3.4 in order to receive credit. You will be submitting Classifier.py to submit.cs.umd.edu. If you chose to use additional examples of sentences and non-sentences, you should instead submit a tar archive that includes Classifier.py and a folder called 'lex' which will include any text files of sentences you used (i.e. everything needed to run your code). Your project will be graded based on how it performs on the test set of sentences provided to you AND a separate test set of sentences that were not released to you.

10% Extra credit for submissions before midnight, Dec 14
5% Extra credit for submissions before midnight, Dec 15
-25% Penalty for submissions before midnight, Dec 17
-50% Penalty for submissions before midnight, Dec 18