Programming Assignment 05: Word Embedding and Random Forest Classifiers (50 points)

In this assignment you will experiment with word embedding (word2vec) and with Random Forest classifiers.

Software

In addition to Python, you will need the following packages**: numpy, sklearn, and gensim**. You may need other libraries too depending on your particular implementation. They should all be easily installable via pip command.

Problem: Sentiment analysis

We will use the same dataset as for the Naïve Bayes for sentiment analysis. Recall that the dataset comes already split into training and testing sets and that the data for each class is saved in separate files. Please refer to PA3 for details. There is also code provided in PA3 for reading and tokenizing the data that you can reuse.

In this assignment we will use vector embedding to represent each document and apply Random Forest classifier to the vector representations.

Gensim

The python package gensim provides pretrained word2vec models. After you install gensim via pip command, you can use the following code to load a pretrained model:

import gensim.downloader as api

wv = api.load('word2vec-google-news-300')

The variable wv contains the model, and you can retrieve a vector for individual word, for instance

wv.get_vector('road')

or you can get the mean vector for a set of words, for instance

wv.get_mean_vector(['a' 'long' 'and', 'winding', 'road'])

Each sample in the sentiment analysis dataset is a sentence, and you will tokenize it and use the mean vector function to get a vector representation of each sentence.

Random Forest

The sklearn package provides many types of classifiers, and we will work with Random Forest in this assignment.After you install sklearn with pip command you need to import the classifier like this:

from sklearn.ensemble import RandomForestClassifier

you also should import the metrics package to evaluate your results:

from sklearn.metrics import accuracy_score

You can initialize the forest like this:

rf = RandomForestClassifier(n_estimators=1000)

and train it like this, where X is the data with second dimension equal to 300 (do you see why?) and y is the labels:

rf.fit(X, y)

and test it like this, XTest is the test data:

y_pred = rf.predict(XTest)

accuracy_score is a useful function to evaluate the accuracy of classification.

You can find more information in documentation for Random Forest classifier here: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Problem.

Part 1: Classifier Training

You need to create training and testing datasets from the texts and train and test Random Forest classifier. Try different values for ***nestimators*** and explain your observations. How do your results compare to the accuracy of Naïve Bayesian classifier in PA3?

Part 2: Accuracy study

The training dataset contains 3097 sentences, 949 positive, 893 negative, and 1255 neutral. What happens if we have a much smaller training dataset? Modify the code for both Naïve Bayes classifier from PA3 and your new classifier to train on a subset of the training dataset. Compute classification accuracy as function of the training set size and explain your results. Use the following size of the training set: 25, 50, 150, 200, 300.

Hint: The samples in the training dataset are ordered by label. You can use numpy.random.permutation function to shuffle the samples and produce a random subset of the training dataset. Don't forget to use the same indexes for the labels and the data samples!

SUBMISSION: Submit the python files and a short report with your results as a PDF file. The code will also be evaluated on readability.