CS 584-A: Natural Language Processing

Homework 1

Due Date: Monday 11:59 AM EST, October 2, 2023 **Total: 100 points**

Goals

The goal of HW1 is for you to get familiar with extracting features and standard machine learning workflows, including reading data, preprocessing data, training, testing, and evaluation. All those are open questions and there is no fixed solution. The difference in data processing, parameter initialization, data split, etc, will lead to the differences in final predications and evaluation results. Therefore, during the grading, the specific values in the results are not important. It is important that you focus on implementing the functions and setting up the pipelines.

Dataset

The dataset is about the product review dataset from Amazon. Please download it from Canvas. The dataset provides product reviews and overall ratings for 4,195 products. We will consider the fine-grained overall ratings as labels, which are ranging from 1 to 5: highly negative, negative, neutral, positive, and highly positive.

Task: Sentiment Analysis with Text Classification

Based on the dataset, the task is to perform sentiment analysis, which predicts the sentiments based on the free-text reviews. You can solve the task as a *multi-class classification problem* (5 classes in total). You can also simplify the task as a *binary classification problem*. Specifically, we will consider the rating 4 and 5 as positive, and rating 1 and 2 as negative. For data samples with neutral rating 3, you can discard them or simply consider them as either positive or negative samples. ***In the submission, please make sure to clearly mention which task you are solving and how did you prepare the class labels.***

Tasks 1: Extracting features (60 points)

1. **Data preparation (20 points)**
   1) **Data preprocessing (10 points)**: Download and load the dataset. Please process the reviews by (i) converting all text to lowercase to ensure uniformity, (ii) removing punctuations, numbers, and stopwords, and (iii) tokenizing the reviews into tokens. If you plan to work on the binary classification problem, you will need to assign binary class labels based on the above-mentioned strategy.
   2) **Data split (5 points)**: Split the data with the ratio of 0.8, 0.1, and 0.1 into training, validation/development, and testing, respectively.
   3) **Data statistics (5 points)**: Please conduct an analysis of the basic statistics of the data you obtained. For example, you can consider the following aspects, number of data samples in training/development/testing, minimum/average/maximum number of tokens across all reviews, number of positive/negative reviews in training/development/testing. You can

also provide other basic statistics of the data if you think they are important. Please report these as the table or plots.

2. **Representation of Texts: word vectors (40 points)**
   You may find this jubyter notebook useful when working on the implementation.
   1) **Count-based word vectors with co-occurrence matrix (20 points, 5 points for each question)**
      a. Please implement a function named ***get_vacab(corpus)*** that returns corpus_words, which is the list of all the distinct words used in the review corpus. You can do this with 'for' loops, but it's more efficient to do it with Python list comprehensions. The returned corpus_words should be sorted. You can use python's sorted function for this.
      b. Based on the word vocabulary obtained with ***get_vacab(corpus)*** function, please implement a function named ***compute_co_occurrence_matrix(corpus, window_size=4)*** that returns both M and word2index. Here, M is the co-occurrence matrix of word counts and word2index is a dictionary that maps word to index. The function constructs a co-occurrence matrix for a certain window-size n (with a default of 4), considering words n before and n after the word in the center of the window. You can use numpy to represent vectors, matrices, and tensors.
      c. Please implement a function named ***reduce_to_k_dim(M)*** performs dimensionality reduction on the matrix M to produce k-dimensional embeddings and returns the new matrix M_reduced. Use SVD (use the implementation of Truncated SVD in sklearn sklearn.decomposition.TruncatedSVD, set n_iters = 10) to take the top k components and produce a new matrix of k-dimensional embeddings.
      d. Implement ***plot_embeddings(M_reduced, word2index, words_to_plot)*** to plot in a scatterplot the embeddings of the words specified in the list 'words_to_plot'. Here, '***M_reduced'*** is the matrix of 2-dimensional word embeddings obtained in question c. word2index is the dictionary that maps words to indices for the embedding matrix obtained in question b.
      Use the implemented function to get the plot for the following list of words words_to_plot=['purchase', 'buy', 'work', 'got', 'ordered', 'received', 'product', 'item', 'deal', 'use'], and show the plot.

   2) **Prediction-based word vectors from Glove (20 points, 5 points for each question)**
      a. Please use the provided ***load_embedding_model()*** function to load the GloVe embeddings. Note: If this is your first time to run these cells, i.e. download the embedding model, it will take a couple minutes to run. If you've run these cells before, rerunning them will load the model without redownloading it, which will take about 1 to 2 minutes.

```
def load_embedding_model():
    """ Load GloVe Vectors
        Return:
            wv_from_bin: All 400000 embeddings, each lengh 200
    """
    import gensim.downloader as api
    wv_from_bin = api.load("glove-wiki-gigaword-200")
```

```
    print("Loaded vocab size %i" % len(list(wv_from_bin.index_to_key)))
    return wv_from_bin


# ----------------------------------
# Run Cell to Load Word Vectors
# Note: This will take a couple minutes
# ----------------------------------
wv_from_bin = load_embedding_model()
```

b. Select the words in the vocabulary returned in 1)a and get the corresponding GloVe vectors. You can adapt the provided function ***get_matrix_of_vectors(wv_from_bin, required_words)*** to select the Glove vectors and put them in a matrix M.

```
def get_matrix_of_vectors(wv_from_bin, required_words):
    """ Put the GloVe vectors into a matrix M.
        Param:
            wv_from_bin: KeyedVectors object; the 400000 GloVe vectors loaded from file
        Return:
            M: numpy matrix shape (num words, 200) containing the vectors
            word2ind: dictionary mapping each word to its row number in M
    """
    import random
    words = list(wv_from_bin.index_to_key)
    print("Shuffling words ...")
    random.seed(225)
    random.shuffle(words)
    words = words[:10000]
    print("Putting %i words into word2ind and matrix M..." % len(words))
    word2ind = {}
    M = []
    curInd = 0
    for w in words:
        try:
            M.append(wv_from_bin.get_vector(w))
            word2ind[w] = curInd
            curInd += 1
        except KeyError:
            continue
    for w in required_words:
        if w in words:
            continue
        try:
            M.append(wv_from_bin.get_vector(w))
            word2ind[w] = curInd
            curInd += 1
        except KeyError:
            continue
    M = np.stack(M)
    print("Done.")
    return M, word2ind
```

c. Use the function ***reduce_to_k_dim()*** you implemented in 1)c to reduce the vectors to 2 dimension. Similar to what you did in 1)c.
d. Use the ***plot_embeddings function in*** 1)d to get the plot for the same set of words in 1)d. Compare the differences of the plot in 1)d and 2)d, provide some analysis, and describe your findings.

Task 2: Sentiment Classification Algorithms (40 points)

3. **Perform sentiment analysis with classification**
   1) **Review embeddings (5 points):** Similar to what you did in 2)c, use the function reduce_to_k_dim() you implemented in 1)c to reduce the vectors to 128 dimension. Based on the word embeddings, get the review embedding by taking the ***average*** of the word embeddings in each review. Write a function for getting review embeddings for each review.
   2) **Models:** Please examine the performance of the following two models on the sentiment analysis task. You can use existing implementations of the models and various packages such as sklearn, Tensorflow, Pytorch, etc.
      a. Logistic regression, with L2 regularization (10 points)
      b. A Neural Network (NN) model for sentiment classification (10 points)
   3) **Evaluation (15 points):** Train the model on the training set, select the best model based on the validation set, and test your model on the testing set.
      a. Evaluate the model performance using metrics for classification, such as accuracy, precision, recall, F1-score, and AUC. Report your results for both methods (10 points).
      b. Have a brief discussion to compare the performance of those two models (5 points). It should be noted that there is no fixed answer for the results. You will need to report the exact results returned in your experiments. The discussions should only base on your own experimental settings and returned results.

Submission guidelines:

1. In your codes, please try to add ***clear comments and detailed steps*** to get all the partial points. Also, include the required discussions and results analysis in the text cell.
2. Please use Jupyter Notebook for your programming. Before submission, please make sure to clear all existing outputs and rerun all the codes. Once it is done to run all the codes, save it as a pdf file. You will need to ***submit both the .ipynb and .pdf files***. Both files should include all the codes, comments, discussions, and answers for the questions.
3. You can also organize all the results and analysis in a separate pdf file. This is optional.
4. Submissions should be made on ***Canvas***.
5. *Each student needs to submit your own answers and codes. Similar or the same answers/codes will not be graded. You will directly fail the course if cheating/plagiarism is found.*
6. ***Properly cite any recourses*** you used for your submission, including but not limited to websites, GitHub repositories, publications, generative AI techniques, etc.