AI534 — Implementation/Experimentation Assignment 4

General instructions.

1. For this assignment, you are free to use sklearn and other packages as specified in the detailed description.

2. You can collaborate in group of up to three people. Please do not share code with other groups, or copy program files/structure from any outside sources like Github. Each group's work should be your own.

3. Submit your report on Canvas and your code on TEACH following this link: `https://teach.engr.oregonstate.edu/teach.php?type=assignment`.

4. Your code should follow the structure and function specification of the provided skeleton code. Your code should assume that the data is in the same folder as the code.

5. Please test your code before submission on the EECS servers (i.e. babylon) to make sure that it runs correctly on the server and produce the correct results as reported in the your report.

6. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. The report should be clear and concise, with figures and tables clearly labeled with necessary legend and captions. The quality of the report and is worth 10 pts. The report should be a PDF document.

7. In your report, the **results should always be accompanied by discussions** of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

Word embeddings and sentiment analysis

(total points: 90 pts + 10 report pts)

The objective for this assignment is twofold. First, you will be playing with unsupervised learning techniques using pre-trained word embeddings. Second, you will have an open ended exploration of using word embeddings to improve the sentiment classification performance.

Data

For the first part, you will be using the GloVe word embedding, which is a popular word embedding that is pre-trained using large amounts of unlabeled text. To avoid having to deal with the full size of the GloVe embeddings, we've provided you with the file GloVe_Embedder_data.txt, containing a reduced subset of GloVe embeddings that corresponds to the intersection of the Twitter sentiment dataset's vocabulary and the full GloVe vocabulary. We've also provided you with GloVe_Embedder.py to handle loading the embeddings from GloVe_Embedder_data.txt as well as embedding vocab elements and finding nearest neighbors. Read the comments in GloVe_Embedder.py for more information. Note that we've used the zero vector to represent unknown vocabulary elements.

For the second part, you will continue working with the natural language sentiment dataset from the last assignment (training set: **IA3-train.csv** and validation set: **IA3-dev.csv**).

Explore word embeddings (50 pts)

Build your own data set of words. (5 pts) You will begin by building a small set of words to visualize and play with. Specifically, use 'flight', 'good', 'terrible', 'help' and 'late' as the initial set of seed word. For each seed word, find 29 most similar words based on the word embedding using Euclidean distance. This will give you a total of 150 words forming five clusters. In the report, please list the 29 most similar words for each seed word.

Dimension reduction and visualization. (25 pts) In this part, you will play with different dimension reduction and visualization techniques.

1. (10 pts) Apply PCA (you can use sklearn.decomposition.pca) to the 150 words and visualize them in 2-d space. In your visualization, you should assign each seed word (and the words similar to that seed word) a distinct color. Do you observe five distinct clusters in your visualization?
2. (15 pts) Next you will apply t-SNE (you can use sklearn.manifold.TSNE with Euclidean distance) to the 150 words and visualize them in 2-d space using the same color mapping. Note that Perplexity is a critical parameter for t-SNE. It is recommended by the authors of t-SNE that the perplexity value should be between 5 and 50. For this assignment, you will need to explore different perplexity values and observe the resulting impact on the visualization. In your report, you are expected to provide substantially different visualization results by using different perplexity parameter.

Clustering. (20 pts) For this part you will apply k-means clustering (you can use sklearn.cluster.kmeans, you can keep most default parameters except for $n\_cluster$) to your words using different k values ranging from 2 to 20. For each k value, record the resulting kmeans objective (or inertia as in sklearn), which measures:

$$\sum_{i=1}^{k} \sum_{x \in C_i} |\mathbf{x} - \mu_i|^2$$

Plot the kmeans objective as a function of k. Do you observe monotonically decreasing objective value as we increase k? Do you see any evidence from this curve that suggests $k = 5$? Provide an explanation for your observations.

Using the original seed word as ground truth labels for clustering, evaluate the clustering solution for different k values using different metrics including:

- Purity (you will need to implement this measure)
- Adjusted rand index (you can use sklearn.metrics.adjusted_rand_score) and
- Normalized Mutual Information (you can use sklearn.metrics.normalized_mutual_info_score).

Plot each metric you get as a function of k. Do you $k = 5$ gives the best score for different metrics? Provide an explanation for your observation. Which of these three metrics are appropriate to use if we are evaluating two different clustering algorithms that automatically search for the number of clusters in the data (that is, one algorithm might find five clusters in the data while the other might find ten)?

Using word embeddings to improve classification (40 pts)

For this part, we will work with the sentiment classification dataset used in IA3 and conduct a somewhat open ended exploration to answer the following question:

> How can you improve the bag-of-words representation for classification using the word embeddings?

Note that we are not looking for deep-learning based approaches. You are still limited to using classifiers that we learn in this class so far. The goal is to leverage the word embeddings to improve the representation for the tweets so that the classification performance can be improved. Here are some basic ideas to get you started.

One simple idea of using the word embedding is to represent a tweet using the weighted average of the embeddings of the words in the tweet. Another possible idea is to cluster the words in the vocabulary and represent each tweet using bag-of-word-clusters. This will substantially reduce the dimension and redundancy of the features.

A third direction is to improve the utilization of the GloVe embeddings. The provided GloVe_Embedder _data.txt simply uses the intersection of the GloVe vocabulary and the dataset's vocabulary, requiring exact matches for an entry to be included in the joint vocabulary. Many entries in the dataset's vocabulary are not in the GloVe vocabulary, leading to many potentially useful features being replaced by the zero vector. You could allow for inexact matches between the dataset vocab and the GloVe vocab. Some examples: you could stem unknown words and match the stem to the GloVe vocab, or you could split unknown words into two parts which are in the GloVe vocab, then use the average of the two GloVe embeddings to represent the unknown word (e.g., represent "americanair" as the average of "american" and "air" embeddings).

If you do change the vocabulary, make sure to store your new vocabulary as a text file in the same format as GloVe_Embedder_data.txt, *call it something other than GloVe_Embedder_data.txt*, and provide your new vocab file along with your submission. Hint: Vocab_Embedder.py has a save_to_file() function, and loads from file on init. Make sure Vocab_Embedder.py can save and load your custom vocab.

You can find the full GloVe vocabulary here: `https://nlp.stanford.edu/data/glove.twitter.27B.zip` Note that GloVe provides embeddings of different dimensions. We used the 200 dimensional GloVe embeddings, glove.twitter.27B.200d.txt. Or, you can use a package like gensim, as demonstrated here: `https://kavita-ganesan.com/easily-access-pre-trained-word-embeddings-with-gensim/`.

Your report should include a clear description of each method you explored, the motivation behind the method and training and validation accuracy of the method.

This dataset has a high validation accuracy achievable using bag-of-words representation (92%+). So it may be very challenging to further improve the performance. Hence, you will be evaluated based on the exploration itself, not necessarily based on the end results. That is, if you explore some interesting ideas that didn't lead to performance improvement, you will still be rewarded.