CSC 495/693: Natural Language Processing, Spring 2024 Assignment 2

Assigned Wednesday Feb 21, due Friday March 8. Max points: 100.

In this assignment, you will practice word embeddings, NLP evaluation and text annotation.

Programming Requirements

- Dataset: You have been provided with 2,000 movie comments as the dataset "comments2k.zip" including 1,000 positive comments and 1,000 negative comments.
- Programming Language: Please use Python3, not Python2.
- Coding Style: define a function for each question.

1. Word Embedding (60 points)

Word embeddings are a form of word representation that bridges the human understanding of language to that of a machine using vectors. Use **all the 2,000 comments** as the corpus, and choose Gensim, GloVe or other related python libraries to finish the following tasks.

1) Train word embeddings vectors using word2vec method with CBOW style (vector_size=100, window=5, min_count=1). Report your computer's parameters (or Google Colab) and the time used for training.
2) Train word embeddings vectors using GloVe method. (vector_size=100, window_size=5, vocab_min_count=1)
3) Evaluate the model trained in 1) and 2) by analyzing the 10 most similar words to each of the following word: *"movie", "music", "woman", "Christmas*". Which model's output looks more meaningful?
4) Train word embeddings vectors using word2vec method with CBOW style and set the vector size as 1, 10, 100 separately. You then have three embedding models *m1, m2* and *m3*.
5) Split the dataset into training, validation and testing sets by 80/10/10. Train a neural network classifier (1 hidden layer, 10 neurons, activation='relu') for the comment sentiment classification. Report the classifier accuracy, precision, recall and F1 score on the testing set given *m1, m2* and *m3* as the pre-trained word embeddings. (Tip: make sure you load the pretrained embeddings) Which one has the best performance? Explain.
6) Suppose we use word2vec to train word vectors with window size as 3. Given a sentence "*Very good drama*", it will be transferred to the training set with instances X and corresponding labels Y. If we choose skip-gram style, what are X and Y in the training set? If we choose CBOW style, what are X and Y in the training set?

2. Text Annotation (40 points)

1) When there is no (enough) labelled corpus to train a machine learning based NLP model, we need to create a training text dataset as golden standard through manual annotation. Choose a text annotation tool to finish the following two text annotation tasks:

   Entity Annotation: "*Barack Obama was the 44th President of the United States. He was born in Hawaii and studied law at Harvard University*."
   Annotation Results: *Barack Obama* PERSON

44th CARDINAL
the United States GPE
Hawaii GPE
Harvard University ORG

Sentiment Annotation: *"De Niro has the ability to make every role he portrays into acting gold. He gives a great performance in this film and there is a great scene where he has to take his father to a home for elderly people because he can't care for him anymore that will break your heart. I will say you won't see much bette acting anywhere*."
Annotation Results: Positive

Report the name of the tool you chose. When you are annotating, make a couple of screenshots and put them into the report. After annotation, you should obtain a file with annotation information from the tool: the annotated data file. Submit the annotated data files for the above two tasks.

2) Active learning is a method to improve annotation efficiency. The following code imitates an active learning process.
   a) What is the purpose of the code between "### below" and "### above"? Replace these code and other necessary code (as few as possible) to implement the active learning method in another strategy. Compare these two strategies, which one is better in this example?
   b) If the code is used for movie review annotation, how many reviews need to be labelled by the annotator every time? Discuss the possible pros and cons by increasing and decreasing this number.

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Generate a synthetic dataset
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)

# Split the dataset into initial training set and pool set
X_train, X_pool, y_train, y_pool = train_test_split(X, y, test_size=0.9, random_state=42

# Initialize the active learning loop
iterations = 10
batch_size = 10
model = LogisticRegression(random_state=42)

for i in range(iterations):
    print("Iteration {}:".format(i+1))

    # Train the model on the current training set
    model.fit(X_train, y_train)

    # Predict the labels of the unlabeled instances in the pool set
    y_pool_pred = model.predict(X_pool)

    ### below
    y_pool_prob = model.predict_proba(X_pool)
    entropy = -np.sum(y_pool_prob * np.log(y_pool_prob), axis=1)
    query_idx = np.argsort(entropy)[-batch_size:]
    ### above
```

```
    X_query = X_pool[query_idx]
    y_query = y_pool[query_idx]

    # Add the labeled instances to the training set and remove them from the pool set
    X_train = np.concatenate([X_train, X_query])
    y_train = np.concatenate([y_train, y_query])
    X_pool = np.delete(X_pool, query_idx, axis=0)
    y_pool = np.delete(y_pool, query_idx)

    # Compute and print the accuracy of the model on the test set
    y_test_pred = model.predict(X_pool)
    accuracy = accuracy_score(y_pool, y_test_pred)
    print("Accuracy: {:.3f}\n".format(accuracy))
```

Writeup

Prepare a writeup on your experiments by using any of the following template:

- ACM (https://www.acm.org/publications/proceedings-template/)
- IEEE (https://www.ieee.org/conferences/publishing/templates.html)

Write down any further insights or observations you made while implementing and running the program. Especially interesting insights may be awarded extra points. You may also receive extra points for well-written code with clear comments and runs efficiently. Conversely, poorly written, or not following the ACM/IEEE format, or hard to understand and inefficient code will lose points.

What to turn in

You will turn in:

1. Your writeup, and
2. Your source code. You may include a readme if needed (e.g. if you wish to bring anything to my attention). Please ensure your code is well documented. **I will not be able to spend a lot of time debugging your code if it crashes during our testing.**

To turn in your code and writeup, use Canvas. Prepare a zip file with all your files and name it <yourname>_assign2.zip. **This zip file should only contain your writeup, source code and readme (if needed) and not executables/object files/data files/unmodified code/anything else, and must be timestamped by the due date to avoid a late penalty.**