CSC 693 - HW 2 - Word2Vec Neural Network

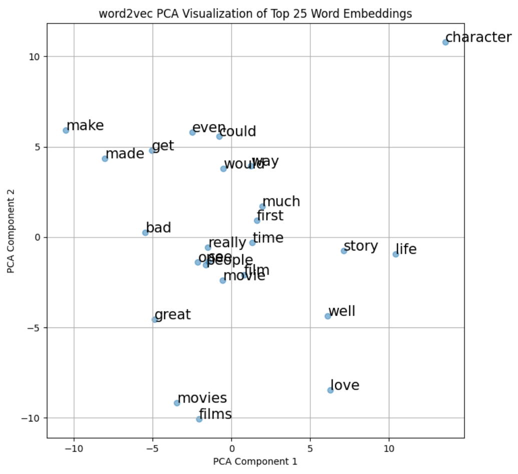
Sahil Dhawan

I. Word Embedding

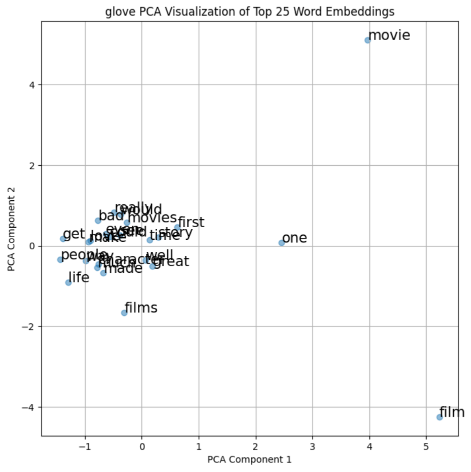
Word embeddings are a form of word representation that bridges the human understanding of language to that of a machine using vectors. Use **all the 2,000 comments** as the corpus, and choose Gensim, GloVe or other related python libraries to finish the following tasks.

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.
   Computer Parameters:
   - Lenovo Yoga 920-13IKB
   - Processor Intel Core i5-8250U @ 1.6GHz × 8
   - Ubuntu 22.04.4 LTS
   - 64-bit

   After being loaded into separate lists, the positive and negative comment data was combined into one list, `all_data`, by appending the negative data after the positive data. `all_data` was preprocessed by tokenizing each comment into words and excluding stop words and punctuation. Using `all_data` and word2vec with CBOW, the time to train 2,000 comments was 5.888 seconds.



2) Train word embeddings vectors using GloVe method. (vector_size=100, window_size=5, vocab_min_count=1) With the same preprocessing as 1), the time to train was 30.933 seconds.



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?

Most Similar Words with word2vec with CBOW (vector_size=100, window=5, min_count=1)							
movie		music		woman		christmas	
film	62.00%	soundtrack	72.29%	maiden	72.19%	carol	75.09%
movies	58.16%	dancing	70.96%	organs	67.64%	ghost	73.29%
stardust	55.11%	songs	68.52%	naive	67.53%	holiday	71.25%
it	54.98%	singing	68.24%	aged	66.78%	croc	65.85%
valentine	54.55%	costumes	66.43%	girl	66.53%	crypt	65.61%
lyrics	54.53%	editing	66.07%	widow	66.37%	festivities	61.17%
slur	54.29%	dance	65.81%	warthog	65.61%	woodward	61.09%
thats	54.23%	noises	65.30%	gigi	65.18%	glib	59.32%
tensed	54.08%	photography	64.87%	herding	65.11%	1984	58.94%
swede	53.64%	rap	64.16%	daughter	64.39%	version	57.72%

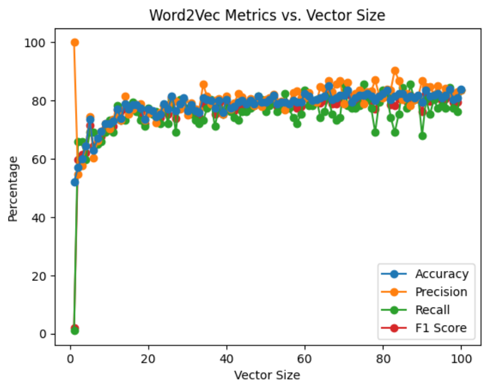
Most Similar Words with GloVe (vector_size=100, window_size=5, vocab_min_count=1							
movie		music		woman		christmas	
kon	91.10%	rap	92.67%	unmarried	91.00%	eve	95.46%
imaging	90.21%	bass	85.82%	fatcheek	89.38%	carol	93.16%
execs	89.57%	dancing	80.38%	laramie	87.11%	ghost	87.02%
confines	89.29%	rrhs	80.20%	isolating	86.92%	present	85.39%
highlander	89.01%	puccini	78.39%	renegade	86.41%	merry	83.05%
suprise	87.22%	emphatic	76.73%	protestant	85.76%	jovial	82.51%
offence	86.50%	dance	75.34%	she	85.30%	spirit	80.20%
organically	86.14%	swelling	74.98%	embittered	83.82%	dickens	75.61%
hartley	84.79%	costumes	71.82%	hollow	83.05%	knotts	72.71%
dedicating	84.57%	quality	71.57%	widow	82.86%	charles	72.62%

GloVe and Word2vec produce outputs that seem to be as meaningful as the other except for certain words. The two most similar words from word2vec for ”movie” are more relevant than the two most similar words from Glove, although ”kon” is part of a Hindi title from a comment describing two Bollywood movies and given the entire code is meant for English, this is likely what threw it off. The similar words for music is better in word2vec with two exceptions: photography and costumes, while GloVe has multiple words that do not necessarily belong such as costumes, swelling, and rrhs (although the comment this token was taken from has rrhs as an acronym for ”Rock and Roll High School”. The similar words for ”Christmas” are overall much more meaningful in GloVe than in the similar words produced by word2vec. As for the similar words to ”woman”, this may be a reflection of the movie reviewers and/or the movies they were reviewing, but many words from both models would be considered mysogynistc descriptions of women such as ”maiden”, ”naive”, and ”aged” in word2vec and ”unmarried”, ”fatcheek”, ”isolating”, ”embittered”, and ”hollow” in GloVe. Regardless, those descriptions are technically meaningful in their relation to the word ”woman”.

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 m_1, m_2, and m_3.

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 m_1, m_2, and m_3 as the pre-trained word embeddings. (Tip: make sure you load the pretrained embeddings) Which one has the best performance? Explain.

Evaluation of Word2Vec Neural Network			
	m_1	m_2	m_3
Accuracy	0.4850	0.6950	0.7900
Precision	0.4850	0.6765	0.7723
Recall	1.000	0.7113	0.8041
F1	0.6532	0.6935	0.7879



m_3 has the best results in all four metrics of the three models. The accuracy, precision, recall, and F1 score all increased with vector size - with an outlier of 100% recall for a vector size of 1 and 100% precision for a vector size close to 1. The size of the vector increasing means there are more dimensions to work with, and more potential relations with other word vectors to be had, allowing for more detailed predictions.

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?

The skip-gram style predicts surrounding words given a target word. In the training set, X and Y would be the following:

```
x: ['very', 'good', 'drama']
y: [['good'], ['very', 'drama'], ['good']]
```

The CBOW style predicts a target word given surrounding words. X and Y would be the following:

```
x: [['good', 'drama'], ['very', 'drama'], ['very', 'good']]
y: ['very', 'good', 'drama']
```

II. Text Annotation

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:

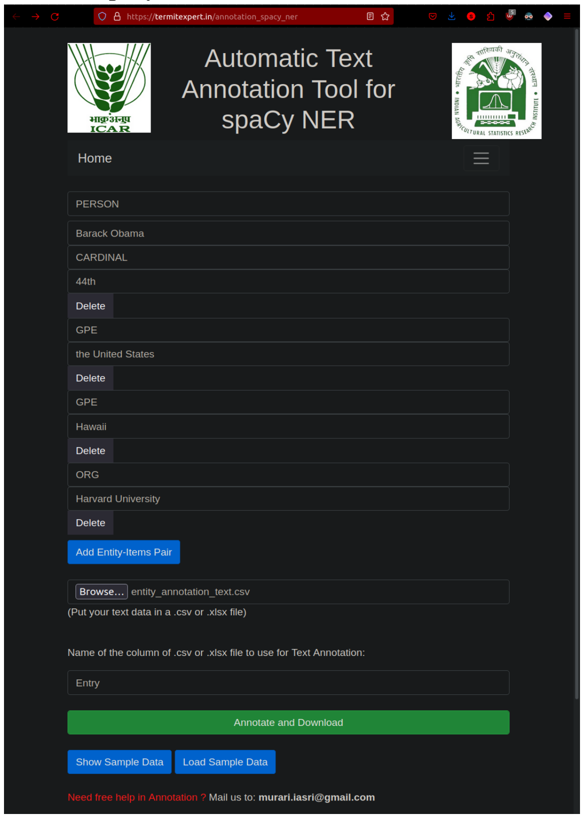
   Entity Annotation:
   "*Barack Obama was the 44th President of the United States. He was born in Hawaii and studied law at Harvard University.*"

   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.

   The tool used for entity annotaion is termitexpert.in/ annotation_spacy_ner


https://termitexpert.in/annotation_spacy_ner

Automatic Text Annotation Tool for spaCy NER

Home

PERSON
Barack Obama
CARDINAL
44th
Delete
GPE
the United States
Delete
GPE
Hawaii
Delete
ORG
Harvard University
Delete
Add Entity-Items Pair

Browse... entity_annotation_text.csv
(Put your text data in a .csv or .xlsx file)

Name of the column of .csv or .xlsx file to use for Text Annotation:
Entry

Annotate and Download

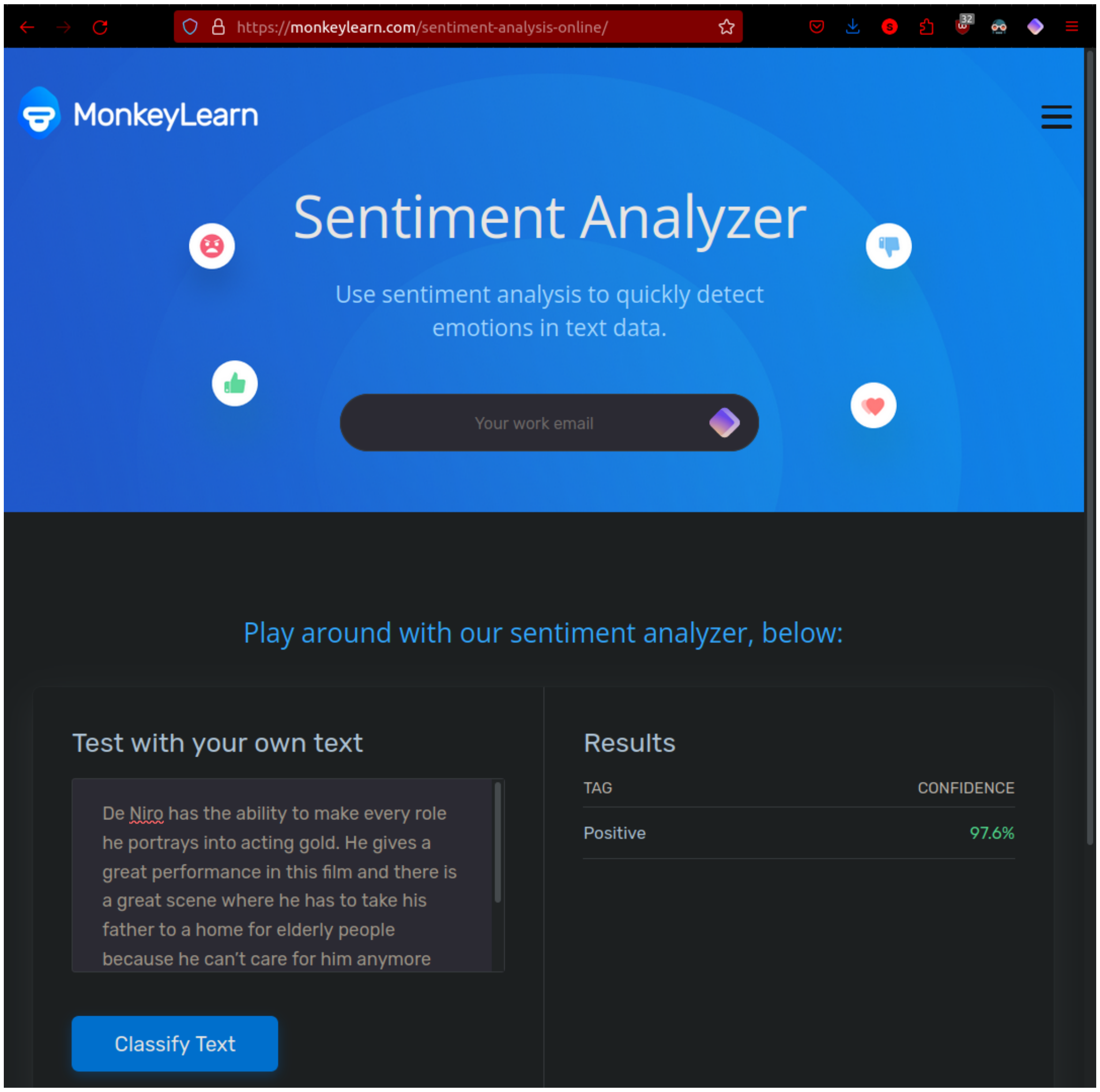
Show Sample Data Load Sample Data

Need free help in Annotation ? Mail us to: murari.iasri@gmail.com

   It produces a text file with the following contents:

```
[('Barack Obama was the 44th President of
    the United States. He was born in Hawaii
    and studied law at Harvard University.
    ', {'entities': [(0, 12, 'PERSON'), (21,
    25, 'CARDINAL'), (73, 79, 'GPE'), (99,
    117, 'ORG')]})]
```

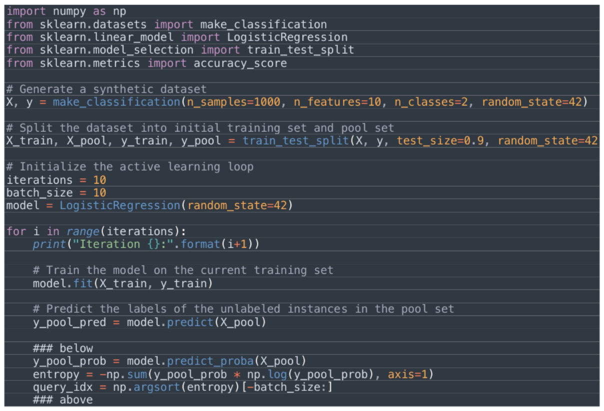
The tool used for sentiment annotation is https://monkeylearn.com/sentiment-analysis-online/. I could not find any tool that gave a file as an output, but it rated the text as positive with 97.6% confidence.


https://monkeylearn.com/sentiment-analysis-online/
MonkeyLearn
Sentiment Analyzer
Use sentiment analysis to quickly detect emotions in text data.
Your work email
Play around with our sentiment analyzer, below:
Test with your own text
Classify Text
Results

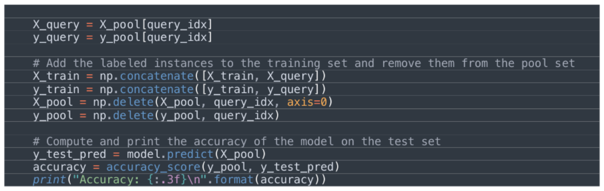
De Niro has the ability to make every role he portrays into acting gold. He gives a great performance in this film and there is a great scene where he has to take his father to a home for elderly people because he can't care for him anymore

TAG	CONFIDENCE
Positive	97.6%

2) Active learning is a method to improve annotation efficiency. The following code imitates an active learning process.



```
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))
```

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?

The purpose of the three lines of code is to prepare to remove the values in `x_pool` and `y_pool` with the largest probability entropies, where entropy is a measure of randomness of the probabilities.

- `y_pool_prob = model.predict_proba(x_pool)`
  This line uses a logistic regression model to predict the probabilites of `x_pool` and stores them in `y_pool_prob`.
- `entropy = -np.sum(y_pool_prob * np.log(y_pool_prob), axis=1)`
  This line calculates the entropy of each value in the `y_pool_prob`.
- `query_idx = np.argsort(entropy)[-batch_size:]`
  `np.argsort` sorts the entropy list numerically and returns the original indices of the sorted values. `[-batch_size]` takes the last n elements, where $n =$ `batch_size`, of the array of indices produced by `np.argsort` and stores those last indices in `query_idx` to be used for selecting elements based on their indices in the next two lines.

replacing the last two lines of code between `### below` and `### above` with:

```
least_confidence = [np.max(y_pool_prob) - np.average(prob_val) for prob_val in
    y_pool_prob]
query_idx = np.argsort(least_confidence)[-batch_size:]
```

calculates the least confidence, or the difference of x probability in the list from the maximum probability, and takes the lowest n (batch size) items in the list to be removed after being set equal to `query_idx`. Surprisingly, the accuracy values produced by each iteration are exactly the same as before the change. Altering the batch size by increasing it has more effect, although an increased batch size with either removal method still produced the exact same accuracies for both methods.

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.

The current batch size is 10, so 10 movie reviews would need to be labelled by the annotator each time. This helps reduce the time spent annotating, but may lower accuracy. If the batch size is increased, more reviews will need to be labelled which is more time consuming on a large scale dataset, but the accuracy would be increase as the batch size increases.