CBOW and Skip gram encoding and evaluation with ML models

Aim: To apply CBOW encoding to the dataset and perform different Machine Learning Models on the dataset.

Description:

The CBOW architecture includes a deep learning classification model that uses context words as input (X) to predict our target word, Y. Considering the following scenario: Have a wonderful day.

Let the word "excellent" be the input to the Neural Network. It's important to note that we're attempting to predict a target word (day) from a single context input word, unique. More specifically, we compare the output error of the one-hot encoding of the input word to the one-hot encoding of the target word (day) .

The context words are predicted in the skip-gram model given a target (center) word. Consider the following sentence: "Word2Vec uses a deep learning model in the backend." Given the center word 'learning' and a context window size of 2, the model tries to predict ['deep,' 'model'], and so on.

```
# For Data Preprocessing
import pandas as pd
# Gensim Libraries
import gensim
from gensim.models import Word2Vec,KeyedVectors
# For visualization of word2vec model
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df=pd.read_csv('/content/drive/MyDrive/NLP/revpre.csv')
df
```

	Unnamed: 0.1	Unnamed: 0	Review	Rating	label	Positive Feedback Count
0	0	0	'absolutely wonderful silky sexy comfortable '	4	1	0
1	1	1	'love dress sooo pretty happened find store im...	5	1	4
2	2	2	'high hopes dress really wanted work initially...	3	0	0
3	3	3	'love love love jumpsuit fun flirty fabulous e...	5	1	0
4	4	4	'this shirt flattering due adjustable front ti...	5	1	6
...	...	...	...	...	...	...
23481	23481	23481	'happy snag dress great price easy slip flatte...	5	1	0
23482	23482	23482	'it reminds maternity clothes soft stretchy sh...	3	1	0
23483	23483	23483	'this fit well top see never would worked im g...	3	0	1
23484	23484	23484	'bought dress wedding summer cute unfortunatel...	3	1	2
23485	23485	23485	'this dress lovely platinum feminine fits perf...	5	1	22

23486 rows × 6 columns

CBOW

```
model_cbow = Word2Vec(sentences=df['Review'], sg=0, min_count=10, workers=4, window =3, epochs = 20)
```

```
WARNING:gensim.models.word2vec:Each 'sentences' item should be a list of words (usually unicode strings). First item here is instea
```

```
import numpy as np
num_features=100
def make_feature_vec(words, model,num_features):
```

```
    # Function to average all of the word vectors in a given paragraph
    feature_vec = np.zeros((num_features,), dtype="float32")
    nwords = 0
    for word in words:
        if word in model.wv.key_to_index:
            feature_vec = np.add(feature_vec, model.wv.get_vector(word))
            nwords += 1
    if nwords > 0:
        feature_vec = np.divide(feature_vec, nwords)
    return feature_vec

def get_avg_feature_vecs(reviews, model, num_features):
    # Function to generate vectors for all movie reviews in a dataset
    counter = 0
    review_feature_vecs = np.zeros((len(reviews), num_features), dtype="float32")
    for review in reviews:
        review_feature_vecs[counter] = make_feature_vec(review, model, num_features)
        counter += 1
    return review_feature_vecs

# Convert the training and test data into fixed-length feature vectors
data_vecs_cb = get_avg_feature_vecs(df['Review'], model_cbow, num_features)
```

ML models on CBOW

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data_vecs_cb, df.Rating, test_size=0.2)


X_train_2d = np.stack(X_train)
X_test_2d = np.stack(X_test)

from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
scaled_train_embed = scaler.fit_transform(X_train_2d)
scaled_test_embed = scaler.transform(X_test_2d)


clf = GaussianNB()
clf.fit(scaled_train_embed, y_train)
from sklearn.metrics import classification_report

y_pred = clf.predict(scaled_test_embed)

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.17      0.03      0.06       179
               2       0.07      0.84      0.13       316
               3       0.12      0.07      0.09       553
               4       0.23      0.04      0.07       992
               5       0.67      0.12      0.20      2658

        accuracy                           0.14      4698
       macro avg       0.25      0.22      0.11      4698
    weighted avg       0.45      0.14      0.15      4698
```

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(scaled_train_embed, y_train)
from sklearn.metrics import classification_report

y_pred = clf.predict(scaled_test_embed)

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.00      0.00      0.00       179
```

```
              2       1.00      0.00      0.01       316
              3       0.25      0.01      0.02       553
              4       0.22      0.03      0.05       992
              5       0.57      0.98      0.72      2658

       accuracy                           0.56      4698
      macro avg       0.41      0.20      0.16      4698
   weighted avg       0.47      0.56      0.42      4698

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(scaled_train_embed, y_train)
y_pred = classifier.predict(scaled_test_embed)
print(classification_report(y_test, y_pred))
```

```
                precision    recall  f1-score   support

              1       0.00      0.00      0.00       179
              2       0.00      0.00      0.00       316
              3       0.00      0.00      0.00       553
              4       0.00      0.00      0.00       992
              5       0.57      1.00      0.72      2658

       accuracy                           0.57      4698
      macro avg       0.11      0.20      0.14      4698
   weighted avg       0.32      0.57      0.41      4698

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(scaled_train_embed, y_train)
y_pred = knn.predict(scaled_test_embed)
print(classification_report(y_test, y_pred))
```

```
                precision    recall  f1-score   support

              1       0.12      0.03      0.05       179
              2       0.15      0.05      0.08       316
              3       0.18      0.10      0.13       553
              4       0.23      0.21      0.22       992
              5       0.59      0.73      0.65      2658

       accuracy                           0.48      4698
      macro avg       0.25      0.23      0.23      4698
   weighted avg       0.42      0.48      0.44      4698
```

SKIPGRAM

```
#skipgram

num_features=100
model_skipgram = Word2Vec(df['Review'], sg=1, min_count=10, workers=4, window =3, epochs = 20,vector_size=num_features)
data_vecs_sg = get_avg_feature_vecs(df['Review'], model_skipgram, num_features)
```

```
WARNING:gensim.models.word2vec:Each 'sentences' item should be a list of words (usually unicode strings). First item here is instea
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data_vecs_cb, df.Rating, test_size=0.2)


X_train_2d = np.stack(X_train)
X_test_2d = np.stack(X_test)
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
scaled_train_embed = scaler.fit_transform(X_train_2d)
scaled_test_embed = scaler.transform(X_test_2d)


clf = GaussianNB()
clf.fit(scaled_train_embed, y_train)
from sklearn.metrics import classification_report

y_pred = clf.predict(scaled_test_embed)

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.07      0.02      0.03       156
               2       0.08      0.84      0.14       330
               3       0.11      0.09      0.10       586
               4       0.30      0.04      0.07       965
               5       0.72      0.11      0.19      2661

        accuracy                           0.14      4698
       macro avg       0.25      0.22      0.11      4698
    weighted avg       0.49      0.14      0.14      4698
```

```
#RandomForest
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier()
clf.fit(scaled_train_embed, y_train)
from sklearn.metrics import classification_report

y_pred = clf.predict(scaled_test_embed)

print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.00      0.00      0.00       156
               2       0.50      0.01      0.01       330
               3       0.23      0.01      0.02       586
               4       0.26      0.04      0.06       965
               5       0.58      0.98      0.73      2661

        accuracy                           0.56      4698
       macro avg       0.31      0.21      0.16      4698
    weighted avg       0.44      0.56      0.43      4698
```

```
#SVM
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(scaled_train_embed, y_train)
y_pred = classifier.predict(scaled_test_embed)
print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.00      0.00      0.00       156
               2       0.00      0.00      0.00       330
               3       0.00      0.00      0.00       586
               4       0.00      0.00      0.00       965
               5       0.57      1.00      0.72      2661

        accuracy                           0.57      4698
       macro avg       0.11      0.20      0.14      4698
    weighted avg       0.32      0.57      0.41      4698

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are i
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(scaled_train_embed, y_train)
y_pred = knn.predict(scaled_test_embed)
print(classification_report(y_test, y_pred))
```

```
                  precision    recall  f1-score   support

               1       0.08      0.04      0.05       156
               2       0.12      0.05      0.07       330
               3       0.13      0.11      0.12       586
               4       0.23      0.23      0.23       965
               5       0.61      0.70      0.66      2661

        accuracy                           0.46      4698
       macro avg       0.23      0.23      0.22      4698
    weighted avg       0.42      0.46      0.44      4698
```

Conclusion:

- SVM with CBOW gave 0.52 accuracy.
- Random Forest Classifier gave accuracy of 0.49
- KNN give accuracy of 0.46

✓ 1s completed at 12:57 AM