Final Project Part 2 - NLP Tasks

Doug Perez

10/23/2022

1. Title: Sentiment Analysis for Academic Reviews

The goal of this project is to perform sentiment analysis of reviews of academic papers written by academic peers.
The dataset to be used is the Paper Reviews Data Set from UC-Irvine: https://archive.ics.uci.edu/ml/datasets/Paper+Reviews

2: Preprocessing

In this section, we will normalize the data, remove stopwords, and split the data into training and test sets. Note: the reviews contain both English and Spanish content. As 382 of the 405 reviews are in Spanish, this analysis will focus exclusively on the Spanish reviews.

In [1]:

```
# Filter out warnings
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Import libraries
import pandas as pd
import json
import nltk
import numpy as np
import text_normalizer as tn
import model_evaluation_utils as meu
import nltk
import textblob
import re

np.set_printoptions(precision=2, linewidth=80)
```

After a couple of iterations of the analysis, I reached a couple of conclusions that helped boost the accuracy of the sentiment predictions. The first is that I am using the Orientation column rather than the Evaluation column to predict the sentiment. Orientation is a somewhat more subjective score based on a reading of the reviewer's comments. The text is then evaluated from -2 to 2 based on how positive or negative the reviewers comments are.

The second adjustment I made was to eliminate those reviews with an orientation score of 0. These reviews are neither positive nor negative, and are thus not easy to include in a binary sentiment classification. This adjustment lowered the number of reviews in the total sample to 278.

In [3]:

```
# Open the json file and read the lines into a variable
with open("./reviews.json", "r", encoding='utf-8') as input_file:
    data = json.load(input_file)

# Create variable 'papers' for the top-level item of the json
papers = data['paper']

# Gather the id of the paper, its preliminary accept or reject decision, the review of the text, and the numeric evaluation
paper_evaulations = [(paper['id'], paper['preliminary_decision'], review['text'], review['evaluation'], review['orientation'])
                       for paper in papers for review in paper['review']
                       if review['lan'] == 'es' and review['text'] and review['orientation'] != '0']

# Print the count of reviews we will be working with
print(f'Number of Spanish language reviews: {len(paper_evaulations)}')

# Convert the list to a dataframe
df = pd.DataFrame(paper_evaulations)
df.columns = ['id','preliminary_decision', 'review','evaluation','orientation']
df.head()
```

```
Number of Spanish language reviews: 278
```

Out[3]:

	id	preliminary_decision	review	evaluation	orientation
0	1	accept	El artículo presenta recomendaciones prácticas...	1	1
1	1	accept	- El tema es muy interesante y puede ser de mu...	1	1
2	2	accept	Se explica en forma ordenada y didáctica una e...	2	1
3	3	accept	Este trabajo propone un nuevo enfoque basado e...	2	1
4	4	accept	Se realiza un trabajo de modelamiento de encri...	2	2

In [4]:

```
# Here we are going to tokenize the text and remove stopwords
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('spanish')

# Next, we define a function for normalizing the text:  making it lowercase, removing punctuation, removing spaces, etc.
def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

normalize_corpus = np.vectorize(normalize_document)
```

In [5]:

```
# Prepare reviews and evaluations (sentiments) for analysis
reviews = np.array(df['review'])
sentiments = np.array(df['orientation'])

# Convert the sentiment from string to int
int_sentiments = [int(sentiment) for sentiment in sentiments]
# print(int_sentiments)

# For ease of classification, convert the score from a range of -2 to 2 to a binary 'positive' or 'negative'.
binary_sentiment = ['positive' if sentiment >= 0 else 'negative' for sentiment in int_sentiments]
print(binary_sentiment[:20])
```

```
['positive', 'positive', 'positive', 'positive', 'positive', 'negative', 'positive', 'positive', 'negative', 'negative', 'nega
tive', 'positive', 'positive', 'negative', 'negative', 'positive', 'positive', 'positive', 'negative', 'negative']
```

In [6]:

```
# Split the data into test and training sets, then display the dimensions of the sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(reviews, binary_sentiment, test_size=0.3, random_state=42)
print(X_train.shape, X_test.shape)
```

```
(194,) (84,)
```

In [7]:

```
# Normalize the reviews datasets
X_train = normalize_corpus(X_train)
X_test = normalize_corpus(X_test)
```

3. Feature Extraction

In this section, we will use the Count Vectorizer and TF-IDF Vectorizer libraries to convert the text of the training and test sets into features.

In [8]:

```
# Use CountVectorizer and TfIdfVectorizer to build features from the reviews
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

# build BOW features on train reviews
cv = CountVectorizer(binary=False, min_df=0.0, max_df=1.0, ngram_range=(1,2))
cv_train_features = cv.fit_transform(X_train)
# build TFIDF features on train reviews
tv = TfidfVectorizer(use_idf=True, min_df=0.0, max_df=1.0, ngram_range=(1,2),
                     sublinear_tf=True)
tv_train_features = tv.fit_transform(X_train)

# transform test reviews into features
cv_test_features = cv.transform(X_test)
tv_test_features = tv.transform(X_test)
```

In [9]:

```
# Print the shape of the features for CV and TV
print('BOW model:> Train features shape:', cv_train_features.shape, ' Test features shape:', cv_test_features.shape)
print('TFIDF model:> Train features shape:', tv_train_features.shape, ' Test features shape:', tv_test_features.shape)
```

```
BOW model:> Train features shape: (194, 19278)  Test features shape: (84, 19278)
TFIDF model:> Train features shape: (194, 19278)  Test features shape: (84, 19278)
```

4. Main Functionality

In this section, we will train the models, use them to predict whether the review is positive or negative, then evaluate the performance of the different models.

Specifically, we will use the Logistic Regression and SGD Classifier models to predict the sentiment of the review.

We will apply the models to both the Bag of Words (BOW) and TF-IDF features.

In [10]:

```
# Import and initialize the Logistic Regression and SGDClassifier models
from sklearn.linear_model import SGDClassifier, LogisticRegression

lr = LogisticRegression(penalty='l2', max_iter=500, C=1)
svm = SGDClassifier(loss='hinge', max_iter=100) # linear support vector machine
```

In [11]:

```
# Run Logistic Regression model on BOW features and display performance
lr_bow_predictions = meu.train_predict_model(classifier=lr,
                                             train_features=cv_train_features, train_labels=y_train,
                                             test_features=cv_test_features, test_labels=y_test)
meu.display_model_performance_metrics(true_labels=y_test, predicted_labels=lr_bow_predictions,
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.7381
Precision: 0.7424
Recall: 0.7381
F1 Score: 0.7398

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.62      0.67      0.65        30
    negative       0.81      0.78      0.79        54

    accuracy                           0.74        84
   macro avg       0.72      0.72      0.72        84
weighted avg       0.74      0.74      0.74        84


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive         20       10
        negative         12       42
```

This first model, linear regression, performed against the Bag of Words features accurately predicted the sentiment of the review about 74% of the time. The model was somewhat more likely to issue a false positive prediction than a false negative.

In [12]:

```
# Run Logistic Regression model on TF-IDF features and display performance
lr_tfidf_predictions = meu.train_predict_model(classifier=lr,
                                               train_features=tv_train_features, train_labels=y_train,
                                               test_features=tv_test_features, test_labels=y_test)
meu.display_model_performance_metrics(true_labels=y_test, predicted_labels=lr_tfidf_predictions,
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.6429
Precision: 0.4133
Recall: 0.6429
F1 Score: 0.5031

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.00      0.00      0.00        30
    negative       0.64      1.00      0.78        54

    accuracy                           0.64        84
   macro avg       0.32      0.50      0.39        84
weighted avg       0.41      0.64      0.50        84


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive          0       30
        negative          0       54
```

Although the accuracy of this model, Linear Regression, run against the TF-IDF Vectorizer features performed only somewhat worse than the Linear Regression model run against Bag of Words, achieving an accuracy of almost 64%, it clearly has an issue with predicting positive reviews.

In [13]:

```
# Run SDGClassifier model on BOW features and display performance
svm_bow_predictions = meu.train_predict_model(classifier=svm,
                                              train_features=cv_train_features, train_labels=y_train,
                                              test_features=cv_test_features, test_labels=y_test)
meu.display_model_performance_metrics(true_labels=y_test, predicted_labels=svm_bow_predictions,
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.7143
Precision: 0.7143
Recall: 0.7143
F1 Score: 0.7143

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.60      0.60      0.60        30
    negative       0.78      0.78      0.78        54

    accuracy                           0.71        84
   macro avg       0.69      0.69      0.69        84
weighted avg       0.71      0.71      0.71        84


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive         18       12
        negative         12       42
```

The SDG Classifier run against the Bag of Words results in a modest drop in prediction accuracy. Just over 71% of the predictions were accurate, with the same number of false negatives as false positives.

In [14]:

```
# Run SDGClassifier model on TF-IDF features and display performance
svm_tfidf_predictions = meu.train_predict_model(classifier=svm,
                                                train_features=tv_train_features, train_labels=y_train,
                                                test_features=tv_test_features, test_labels=y_test)
meu.display_model_performance_metrics(true_labels=y_test, predicted_labels=svm_tfidf_predictions,
                                      classes=['positive', 'negative'])
```

```
Model Performance metrics:
------------------------------
Accuracy: 0.7857
Precision: 0.8041
Recall: 0.7857
F1 Score: 0.7654

Model Classification report:
------------------------------
              precision    recall  f1-score   support

    positive       0.88      0.47      0.61        30
    negative       0.76      0.96      0.85        54

    accuracy                           0.79        84
   macro avg       0.82      0.71      0.73        84
weighted avg       0.80      0.79      0.77        84


Prediction Confusion Matrix:
------------------------------
                 Predicted:
                   positive negative
Actual: positive         14       16
        negative          2       52
```

The last model, the SDG Classifier run against the TF-IDF features the most accurate model with an accuracy of just under 79%. This model featured the fewest false positives; however, it also featured the most false negatives (except for the clearly flawed SDG Classifier on TF-IDF), perhaps indicating that for sentiment analysis, this model is overly pessimistic.

5. Personal Contribution Statement

As this is an individual project, the work is entirely my own. I will therefore use this section to reflect on the work and discuss potential improvements.

The accuracy of my predictions could be better. I made several iterations of the analysis to achieve the accuracy displayed here. First, I over simplified the sentiment score as a binary representation of positive or negative when the papers were actually evaluated from -2 to 2, with 0 clearly being a neutral score. I chose to include 0 as a positive score. I decided a better approach would be to eliminate the 0 evaluations and doing so would boost accuracy.

I then saw improved performance when I used the orientation column rather than the evaluation column. Orientation was a subjective score of how positive or how negative a user's review actually seemed. Using Orientation yielded more accurate predictions.

The other factor that could affect the accuracy of the predictions is that this is my first attempt at sentiment analysis in a language other than English. Although I used the Spanish language library in Spacy and Spanish stop words, I fear that I may need more time and experience working with NLP in another language before attempting sentiment analysis.

In [ ]: