AAI 520 - Assignment 2

Ricardo Barbosa

Semptember 16, 2024

1 Loading the Dataset into Pandas Dataframe

[1]:
```
import pandas as pd
from sklearn.datasets import fetch_20newsgroups

# load dataset
newsgroups_data = fetch_20newsgroups(subset='all', remove=('headers',␣
↪'footers', 'quotes'))
df = pd.DataFrame({'text': newsgroups_data.data, 'target': newsgroups_data.
↪target})

# display the first 5 rows
print(df.head())
```

```
                                                text  target
0  \n\nI am sure some bashers of Pens fans are pr…      10
1  My brother is in the market for a high-perform…       3
2  \n\n\n\n\tFinally you said what you dream abou…      17
3  \nThink!\n\nIt's the SCSI card doing the DMA t…       3
4  1)    I have an old Jasmine drive which I cann…       4
```

The dataset contains newsgroup documents categorized into 20 different topics, and is loaded into a DataFrame.

2 Preprocessing the Text Data

Next step cleans the text data by removing stopwords, punctuation, and non-alphabetical characters, and converting to lowercase. nltk for stopwords and re for regular expressions is also used to clean the text.

[2]:
```
import numpy as np
import re
import string
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
def preprocess_text(text):
    # to_lowercase
    text = text.lower()

    # remove punct and non-alphanum chars
    text = re.sub(f'[{string.punctuation}]', '', text)

    # remove stop words
    text = ' '.join([word for word in text.split() if word not in stop_words])

    return text

# preprocessing to the dataset
df['cleaned_text'] = df['text'].apply(preprocess_text)

# display clean text
print(df['cleaned_text'].head())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/bandito/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
0    sure bashers pens fans pretty confused lack ki…
1    brother market highperformance video card supp…
2    finally said dream mediterranean new area grea…
3    think scsi card dma transfers disks scsi card …
4    1 old jasmine drive cannot use new system unde…
Name: cleaned_text, dtype: object
```

- nltk to download and apply stopwords.
- preprocess_text function converts characters to lowercase, removes punctuation, and excludes stopwords.1
- Store the cleaned text in a new column called cleaned_text.

3 Splitting the Data into Training and Testing Sets

Next, the data is split into training and testnig sets to evaluate performance later in the process, using of train_test_split from sklearn.

[3]:
```
from sklearn.model_selection import train_test_split

# split the data
X_train, X_test, y_train, y_test = train_test_split(df['cleaned_text'],␣
↪df['target'], test_size=0.2, random_state=42)
```

- Data was split so that 80% will be used for training and 20% for testing.
- The train_test_split function shuffles the data and randomizes distribution of samples between training and testing sets.

- random_state=42 is used to aid with reproducing by others.

4 Using Pre-Trained Word Embedding

Train Word2Vec embeddings using the cleaned text data.

[4]:
```python
from gensim.models import Word2Vec

# tokenize text
X_train_tokens = [text.split() for text in X_train]

# train Word2Vec model
word2vec_model = Word2Vec(sentences=X_train_tokens, vector_size=100, window=5,␣
 ↪min_count=5, workers=4)
word2vec_model.train(X_train_tokens, total_examples=len(X_train_tokens),␣
 ↪epochs=10)

# get the embeddings
def get_average_word2vec(text, model, embedding_dim=100):
    words = text.split()
    word_vectors = [model.wv[word] for word in words if word in model.wv]
    if len(word_vectors) == 0:
        return np.zeros(embedding_dim)
    return np.mean(word_vectors, axis=0)

X_train_embeddings = np.array([get_average_word2vec(text, word2vec_model) for␣
 ↪text in X_train])
X_test_embeddings = np.array([get_average_word2vec(text, word2vec_model) for␣
 ↪text in X_test])
```

- Each document is tokenized by splitting it into words. Word2Vec requires tokenized text as input.
- The Word2Vec model is trained on the tokenized training data. "gensim" gives the ability to specify the size of the word vectors (vector_size=100), the context window (window=5), and the minimum word count. min_count=5 means that words appearing less than 5 times are to be disregarded.
- Once the model is trained, numerical representation is done by averaging word vectors for all words. If no words in a document are in the model's vocabulary then a zero is used.

5 Building and Training a Classification Model

Logistic Regression will classify the newsgroups.

[5]:
```python
from sklearn.linear_model import LogisticRegression

# train a Logistic Regression model
```

```
clf = LogisticRegression(max_iter=1000)
clf.fit(X_train_embeddings, y_train)
```

[5]:
```
LogisticRegression(max_iter=1000)
```

- Logistic Regression is used along with with a high max_iter to help with convergence.
- The model is trained using word embeddings as features and newsgroup categories as labels.

6 Making Predictions on the Test Set

The trained model is used it to predict the newsgroup categories of test data.

[6]:
```
# predict test set categories
y_pred = clf.predict(X_test_embeddings)

# show first 10
print(y_pred[:10])  # Show the first 10 predictions
```

```
[ 9  2 14 18 14  8  2 19 14  7]
```

- The first 10 predictions are printed out

7 Model Performance Evaluation

Evaluate the model's performance using several metrics: Accuracy, Precision, Recall, F1 Score, Confusion Matrix, and AUC-ROC.

[7]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, confusion_matrix, roc_auc_score

# accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# precision, recall, F1-Score
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Precision (weighted): {precision}')
print(f'Recall (weighted): {recall}')
print(f'F1 Score (weighted): {f1}')

# confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n{conf_matrix}')

# AUC-ROC
```

```python
y_pred_prob = clf.predict_proba(X_test_embeddings)
auc_roc = roc_auc_score(y_test, y_pred_prob, multi_class='ovr')
print(f'AUC-ROC: {auc_roc}')
```

```
Accuracy: 0.5809018567639257
Precision (weighted): 0.5785242557989796
Recall (weighted): 0.5809018567639257
F1 Score (weighted): 0.5764987438311348
Confusion Matrix:
[[ 64   3   1   1   0   1   0   4   5   2   2   1   0   6   5  22   3  11
   10  10]
 [  1 102  20   7   3  18   4   9   0   3   0   7  10   6   9   0   2   0
    0   1]
 [  2  26 100  21   5  18   4   8   2   0   1   1   4   1   1   0   0   1
    0   0]
 [  0  11  19  94  28   3   8   2   0   0   0   0  13   2   0   2   0   1
    0   0]
 [  1   8   9  36  98   2   7  15   5   0   0   3  13   2   4   0   0   1
    0   1]
 [  0  22  18   4   0 156   0   3   3   1   0   1   4   0   2   0   0   0
    1   0]
 [  0   1   1  18  12   1 120   9   1   4   2   3  13   0   5   0   3   0
    0   0]
 [  2   1   1   1   3   1   5 120  35   0   1   1   4   5   8   0   4   0
    4   0]
 [  1   0   0   1   2   1   9  28  95   5   1   3   4   3   1   0   5   3
    6   0]
 [  5   0   0   0   0   0   3  16   5 133  32   0   1   1   4   2   1   4
    2   2]
 [  3   2   0   1   0   0   4   7   5  33 135   1   1   1   0   3   2   0
    0   0]
 [  2   7   1   0   2   1   2   1   2   2   0 136  12   3   5   1   7   5
   12   0]
 [  0   5   6  11  10   3  11  15   8   4   0   5 103   4  12   2   0   1
    1   1]
 [  5   4   0   0   1   0   2   5   8   2   1   0   5 149   6   2   0   1
    2   1]
 [  1   4   0   1   0   3   1  15   0   2   3   3   8   6 132   1   4   3
    2   0]
 [ 21   0   1   0   0   0   0   4   2   1   0   1   0   3   1 151   1   6
    2   8]
 [  1   0   0   0   0   0   1   9   6   1   1   4   2   4   4   3 105   4
   34   9]
 [  7   0   1   0   1   0   0   7   4   4   2   5   0   1   6   5   1 122
   10   6]
 [ 13   0   0   0   0   0   2   7   5   0   4   3   1   4   7   3  29  11
   65   5]
 [ 27   1   0   0   1   0   0   4   4   4   1   2   0  12   2  50   7   4
```

```
    7   10]]
AUC-ROC: 0.9372302362139326
```

- Metrics computed to evaluate the performance of the model.
- The confusion matrix serves as visual to see how the model is performing.
- We calculate the AUC-ROC score for multi-class classification.

8 Insights and Future Development

Instead of relying on a train-test split, one might consider using a train-validation-test split, where validation set is used for model selection and hyperparameter tuning. Cross-validation can help the model generalize across different subsets of the data.

If some stopwords (e.g., technical terms or common newsgroup jargon) are important for classification, one can create a custom stopword list that includes domain-specific stopwords. This allows for a more fine-tuned preprocess.

In some cases, rare words might contain valuable information, especially in niche categories. Word2Vec's min_count=5 parameter filters out rare words, but one might experiment with lowering or increasing this threshold depending on the dataset.

If one observes that some categories are underrepresented in the dataset, one can use techniques like SVM or logistic regression to give more importance to minority classes during training. This can help improve performance for underrepresented categories.

9 References

Bengfort, B., Bilbro, R., & Ojeda, T. (2018). *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning.* O'Reilly Media, Inc.
Link: https://www.oreilly.com/library/view/applied-text-analysis/9781491963035/

Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed.). Pearson.
Link: https://web.stanford.edu/~jurafsky/slp3/

Google Developers. (n.d.). ROC and AUC. *Machine Learning Crash Course.*
Link: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

KDnuggets. (2022). Logistic Regression for Classification: Basics and Beyond.
Link: https://www.kdnuggets.com/2022/04/logistic-regression-classification.html

Scikit-learn. (2023). *LogisticRegression*: Scikit-learn Documentation.
Link: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html

Rehurek, R. (2023). Word2Vec Tutorial. *Gensim Documentation.*
Link: https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html#sphx-glr-auto-examples-tutorials-run-word2vec-py