NLP

March 2, 2023

[ ]:

[20]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
import numpy as np
import gensim
from gensim.models import Word2Vec
from gensim.utils import simple_preprocess
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

[ ]:

1 1. Modelo Bag of Word

El modelo BoW (Bag-of-Words) es un modelo muy utilizado en el campo del Procesamiento del Lenguaje Natural (NLP) para representar el texto en una forma que pueda ser procesada por algoritmos de aprendizaje automático.

Básicamente, el modelo BoW convierte un texto en una matriz de frecuencias de las palabras que aparecen en él. En este modelo, se considera cada palabra individualmente y se ignora el orden en que aparecen en el texto. El nombre "Bag-of-Words" (en español, "Bolsa de Palabras") hace referencia a que se trata de una bolsa donde todas las palabras del texto son mezcladas y contadas sin importar su orden.

Para crear un modelo BoW, se sigue los siguientes pasos:

Se crea un vocabulario de todas las palabras únicas en el conjunto de datos de entrenamiento.

Se transforma cada documento de texto en un vector de características. Para cada documento, se cuenta el número de veces que aparece cada palabra del vocabulario en el documento y se crea un vector de características con estos valores.

Se entrena el modelo utilizando los vectores de características de cada documento.

Una vez creado el modelo BoW, se puede utilizar para tareas de clasificación de texto, como por ejemplo, clasificación de sentimientos, clasificación de temas, detección de spam, entre otros.

[2]:
```
# Carga el conjunto de datos de IMDB
data = pd.read_csv('IMDB Dataset.csv')
data
```

[2]:

	review	sentiment
0	One of the other reviewers has mentioned that …	positive
1	A wonderful little production. The…	positive
2	I thought this was a wonderful way to spend ti…	positive
3	Basically there's a family where a little boy …	negative
4	Petter Mattei's "Love in the Time of Money" is…	positive
…	…	…
49995	I thought this movie did a down right good job…	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di…	negative
49997	I am a Catholic taught in parochial elementary…	negative
49998	I'm going to have to disagree with the previou…	negative
49999	No one expects the Star Trek movies to be high…	negative

[50000 rows x 2 columns]

[4]:
```
# Creamos una variable con el texto corregido

data['review'] = data['review'].str.replace('[^\w\s]','').str.lower()
data
```

```
<ipython-input-4-b1cadfcdd288>:3: FutureWarning: The default value of regex will
change from True to False in a future version.
  data['review'] = data['review'].str.replace('[^\w\s]','').str.lower()
```

[4]:

	review	sentiment
0	one of the other reviewers has mentioned that …	positive
1	a wonderful little production br br the filmin…	positive
2	i thought this was a wonderful way to spend ti…	positive
3	basically theres a family where a little boy j…	negative
4	petter matteis love in the time of money is a …	positive
…	…	…
49995	i thought this movie did a down right good job…	positive
49996	bad plot bad dialogue bad acting idiotic direc…	negative
49997	i am a catholic taught in parochial elementary…	negative
49998	im going to have to disagree with the previous…	negative
49999	no one expects the star trek movies to be high…	negative

[50000 rows x 2 columns]

[5]:
```
# Crea el modelo de Bolsa de Palabras
# Son palabra en ingles que se usan para crear patrones
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(data['review']) # vectorizamos sobre la columna␣
 ↪creada
```

[6]:
```
# Imprime el vocabulario y la matriz de términos
print(vectorizer.vocabulary_)
print(X.toarray())
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)


[[0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 …
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]
 [0 0 0 … 0 0 0]]
```

[ ]:
```

```

1.1 1.1 Análisis de clasificiación

Para clasificar automáticamente el texto, utilizaremos el modelo de Bolsa de Palabras que creamos anteriormente junto con el algoritmo de Regresión Logística. La regresión logística es un algoritmo de clasificación que funciona bien con datos de alta dimensionalidad como los que se obtienen del modelo de Bolsa de Palabras.

[8]:
```
# Dividimos el df en test y train

train_size = int(len(data) * 0.8)
X_train = X[:train_size]
X_test = X[train_size:]
y_train = data['sentiment'][:train_size]
y_test = data['sentiment'][train_size:]
```

[9]:
```
# Entrenamos el modelo
```

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
/Users/adrian_gr/opt/anaconda3/lib/python3.8/site-
packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[9]:
```
LogisticRegression()
```

[11]:
```
# Hacemos la evaluación del modelo

# Realiza la predicción en el conjunto de prueba
y_pred = lr.predict(X_test)

# Calcula la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print('Precisión: ', accuracy)
```

```
Precisión:  0.8853
```

1.2 1.1 Análisis de Sentimiento

Para realizar un análisis de sentimiento, utilizaremos el modelo de Bolsa de Palabras junto con el algoritmo de Clasificación Naive Bayes. Naive Bayes es un algoritmo de aprendizaje supervisado que se utiliza comúnmente para el análisis de sentimiento. En este caso, utilizaremos la implementación del algoritmo de Clasificación Naive Bayes que se encuentra en la biblioteca sklearn.

[16]:
```
# Entrenamos el modelo naibe
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

[16]:
```
MultinomialNB()
```

[18]:
```
# Evaluamos el modelo
# Realiza la predicción en el conjunto de prueba
y_pred = nb.predict(X_test)

# Calcula la precisión del modelo
accuracy = accuracy_score(y_test, y_pred)
print('Precisión: ', accuracy)
```

```
Precisión:  0.8582
```

[ ]:

2 1. Modelo Bag of Word

El modelo Word2Vec (W2V) es un modelo de lenguaje natural utilizado para crear representaciones vectoriales de palabras en un espacio de alta dimensión. Fue desarrollado por Tomas Mikolov y su equipo en Google en 2013. El modelo utiliza una red neuronal para aprender vectores de palabras a partir de grandes cantidades de datos de texto no etiquetados.

La idea detrás del modelo W2V es que las palabras que se usan en contextos similares tendrán vectores de palabras similares. Por lo tanto, al aprender las representaciones vectoriales de palabras, el modelo es capaz de capturar la similitud semántica entre las palabras.

Existen dos tipos principales de modelos W2V:

CBOW (Continuous Bag of Words): este modelo utiliza un conjunto de palabras de contexto para predecir una palabra objetivo. Por ejemplo, si el contexto es "La ___ está caliente", el modelo puede predecir la palabra "comida" como la palabra objetivo.

Skip-gram: este modelo utiliza una palabra objetivo para predecir un conjunto de palabras de contexto. Por ejemplo, si la palabra objetivo es "comida", el modelo puede predecir las palabras de contexto como "deliciosa", "caliente", "fría", etc.

Una vez que se han aprendido las representaciones vectoriales de las palabras, se pueden utilizar para una variedad de tareas en procesamiento del lenguaje natural, como la traducción automática, la clasificación de texto y el análisis de sentimientos. Además, estos vectores también se pueden visualizar en un espacio de baja dimensión para ayudar a comprender las relaciones semánticas entre las palabras.

En resumen, el modelo Word2Vec es un modelo de lenguaje natural que aprende representaciones vectoriales de palabras utilizando una red neuronal. Estas representaciones vectoriales se utilizan para capturar similitudes semánticas entre las palabras y se pueden utilizar en una variedad de tareas de procesamiento de lenguaje natural.

[25]:
```
# Cargamos los datos de ejemplo

data = ['Este es el primer documento.', 'Este es el segundo documento.', 'Este␣
 ↪es el tercer documento.', 'Este es el cuarto documento.']

# Preprocesamiento de texto: dividimos el texto por palabra
corpus = []
for i in range(len(data)):
    text = data[i].lower()
    words = nltk.word_tokenize(text)
    corpus.append(words)

corpus
```

[25]:
```
[['este', 'es', 'el', 'primer', 'documento', '.'],
 ['este', 'es', 'el', 'segundo', 'documento', '.'],
 ['este', 'es', 'el', 'tercer', 'documento', '.'],
 ['este', 'es', 'el', 'cuarto', 'documento', '.']]
```

[27]:
```
# Entrena el modelo Word2Vec

model = Word2Vec(corpus, window=5, min_count=1, workers=4)
```

[28]:
```
# Crea una matriz de características utilizando los vectores de palabras
X = []
for text in corpus:
    vector = np.zeros(100)
    for word in text:
        vector += model[word]
    X.append(vector)
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-28-7f82f69b4984> in <module>
      4     vector = np.zeros(100)
      5     for word in text:
----> 6         vector += model[word]
      7     X.append(vector)


TypeError: 'Word2Vec' object is not subscriptable
```

[33]:
```
import numpy as np
import gensim
from gensim.models import Word2Vec
import nltk
nltk.download('punkt')

# Cargamos los datos de ejemplo
data = ['Este es el primer documento.', 'Este es el segundo documento.', 'Este␣
 ↪es el tercer documento.', 'Este es el cuarto documento.']

# Preprocesamiento de texto: dividimos el texto por palabra
corpus = []
for i in range(len(data)):
```

```
    text = data[i].lower()
    words = nltk.word_tokenize(text)
    corpus.append(words)

# Entrenamos el modelo Word2Vec
model = Word2Vec(corpus, vector_size=100, window=5, min_count=1, workers=4)

# Creamos una matriz de características utilizando los vectores de palabras
X = []
for i in range(len(corpus)):
    vector = np.zeros(100)
    for word in corpus[i]:
        vector += model[word]
    X.append(vector)

# Creamos una lista de etiquetas (0 para los primeros dos documentos y 1 para␣
↪los últimos dos)
y = [0, 0, 1, 1]

# Entrenamos un modelo de clasificación Naive Bayes
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X, y)

# Realizamos predicciones en los datos de prueba
y_pred = clf.predict(X)

# Imprimimos las predicciones
print("Predicciones:", y_pred)

# Realizamos un análisis de sentimiento
pos_count = 0
neg_count = 0
for i in range(len(corpus)):
    vector = np.zeros(100)
    for word in corpus[i]:
        if word in model.wv:
            vector += model.wv[word]
    prediction = clf.predict([vector])[0]
    if prediction == 0:
        neg_count += 1
    else:
        pos_count += 1
print("Análisis de sentimiento:")
print("Positivo:", pos_count)
print("Negativo:", neg_count)
```

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

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)

<ipython-input-33-370d3a6ede8b> in <module>
 23     vector = np.zeros(100)
 24     for word in corpus[i]:
---> 25         vector += model[word]
 26     X.append(vector)
 27

TypeError: 'Word2Vec' object is not subscriptable
```

[ ]: