TIC

Lección 2: Uso de palabras de parada

TIC

Desarrollo de la sesión

El concepto de palabras de parada o stopwords en representa un área fundamental en la preprocesamiento de texto, con implicaciones significativas en la calidad y eficiencia de los análisis posteriores. Profundizar en este tema implica entender no solo cómo se identifican y eliminan estas palabras, sino también cómo su presencia o ausencia puede afectar los resultados de las tareas de NLP.

En primer lugar, las palabras de parada son aquellas que aparecen con alta frecuencia en un texto pero que aportan poco significado semántico o contexto importante para comprender el contenido real. Estas palabras suelen incluir artículos, preposiciones, conjunciones y otras palabras comunes que se encuentran en todos los textos, independientemente del tema o dominio. Ejemplos típicos en inglés incluyen "the", "and", "of", "to", entre otros.

La eliminación de palabras de parada es una etapa esencial en el preprocesamiento de texto porque ayuda a reducir el ruido en los datos y a enfocar el análisis en las palabras más relevantes y significativas. Al eliminar estas palabras, se simplifica el vocabulario y se destaca el contenido verdaderamente informativo del texto, lo que facilita tareas posteriores como la clasificación de documentos, el análisis de sentimientos o la extracción de información.

Sin embargo, es importante destacar que la lista de palabras de parada no es universal y puede variar según el idioma, el dominio de aplicación y los objetivos específicos del análisis. Por ejemplo, en ciertos contextos como el análisis de sentimientos en redes sociales, palabras como "not" o "but" pueden ser cruciales para determinar la polaridad de una oración y no deberían ser eliminadas como stopwords.

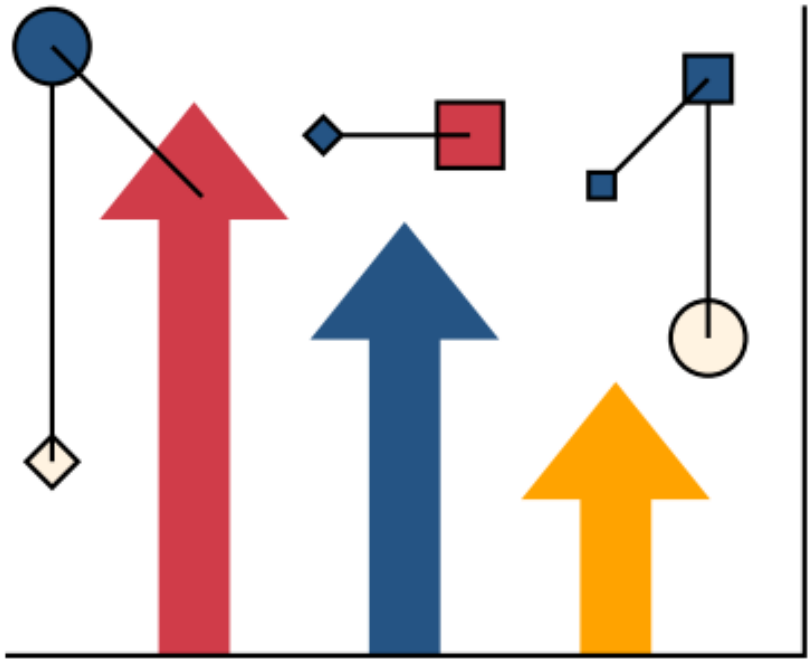
Además, en algunos casos, la eliminación de palabras de parada puede no ser apropiada o incluso contraproducente. Por ejemplo, en tareas de análisis de tema donde las palabras de parada pueden ser indicadores importantes de los temas tratados en un documento, su eliminación podría llevar a una pérdida de información relevante.

Además, en algunos casos, la eliminación de palabras de parada puede no ser apropiada o incluso contraproducente. Por ejemplo, en tareas de análisis de tema donde las palabras de parada pueden ser indicadores importantes de los temas tratados en un documento, su eliminación podría llevar a una pérdida de información relevante.

Ejemplo en Python

```python
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


# Descargar palabras de parada si es necesario
nltk.download('stopwords')


# Definir texto de ejemplo
texto = "El análisis de texto es una técnica fundamental en el procesamiento de lenguaje natural."


# Tokenizar el texto
tokens = word_tokenize(texto)


# Obtener palabras de parada en español
stop_words = set(stopwords.words('spanish'))
```

```
# Filtrar palabras de parada del texto

filtered_text = [word for word in tokens if word.lower() not in stop_words]


# Imprimir resultado

print("Texto original:")

print(texto)

print("\nTexto sin palabras de parada:")

print(' '.join(filtered_text))
```



El script realiza actividades desde la importación de nltk, algun texto de entrada , tokenización y la eliminación de palabras de parada, observe la salida del script anterior:

Texto original:
El análisis de texto es una técnica fundamental en el procesamiento de lenguaje natural.

Texto sin palabras de parada:
análisis texto técnica fundamental procesamiento lenguaje natural .

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

Al relacionar el script con la explicación, se observa que la línea de código stop_words = set(stopwords.words('spanish')) descarga y almacena la lista de palabras de parada en español. Esta lista se utiliza luego para filtrar las palabras del texto tokenizado en la línea filtered_text = [word for word in tokens if word.lower() not in stop_words].

Ejemplo de uso:

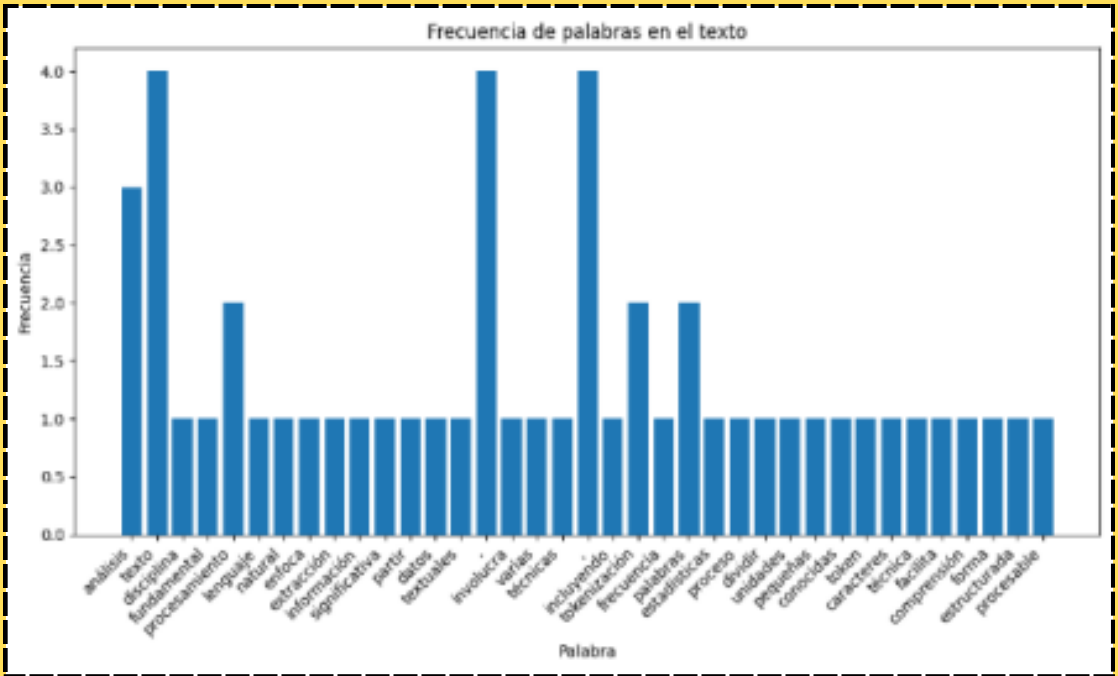
```
import nltk

from nltk.tokenize import word_tokenize

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from collections import Counter


# Descargar recursos de NLTK si es necesario

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')


# Definir texto de ejemplo

texto = """

El análisis de texto es una disciplina fundamental en el procesamiento del
lenguaje natural que se enfoca en la extracción de información
significativa a partir de datos textuales. Este análisis involucra varias
técnicas, incluyendo tokenización, análisis de frecuencia de palabras y
estadísticas de texto. La tokenización es el proceso de dividir un texto
en unidades más pequeñas, conocidas como tokens, como palabras o
caracteres. Esta técnica facilita la comprensión y el procesamiento del
texto de forma estructurada y procesable.

"""


# Tokenización

tokens = word_tokenize(texto)


# Eliminación de palabras de parada
```

```
stop_words = set(stopwords.words('spanish'))

tokens_filtrados = [word for word in tokens if word.lower() not in
stop_words]


# Lematización

lemmatizer = WordNetLemmatizer()

tokens_lematizados = [lemmatizer.lemmatize(word) for word in
tokens_filtrados]


# Análisis de frecuencia de palabras

frecuencia_palabras = Counter(tokens_lematizados)


# Mostrar resultados

print("Texto original:")

print(texto)

print("\nTokens:")

print(tokens)

print("\nTokens filtrados (sin palabras de parada):")

print(tokens_filtrados)

print("\nTokens lematizados:")

print(tokens_lematizados)

print("\nFrecuencia de palabras:")

print(frecuencia_palabras)
```

El anterior script más robusto que aborda varios aspectos del procesamiento de lenguaje natural, incluyendo tokenización, eliminación de palabras de parada, lematización y análisis de frecuencia de palabras.

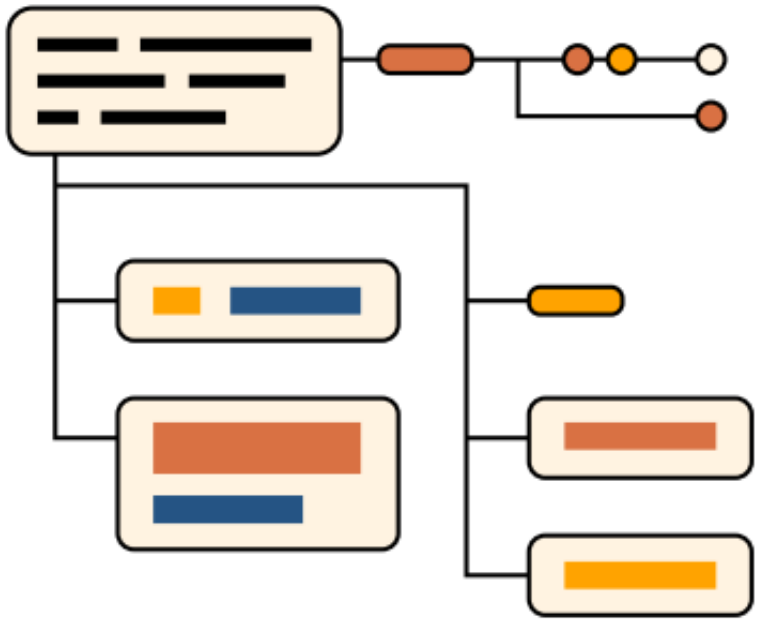
Si se quisiera graficar la frecuencia de cada palabra después de la lematización y la eliminación de palabras de parada en el texto. Se añade la siguiente sesión de código:

```
# Crear gráfico de barras de frecuencia de palabras

plt.figure(figsize=(10, 6))

plt.bar(frecuencia_palabras.keys(),
frecuencia_palabras.values())

plt.title('Frecuencia de palabras en el texto')

plt.xlabel('Palabra')

plt.ylabel('Frecuencia')

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.show()
```



Esta muestra la distribución de palabras según su frecuencia importante para verificar el impacto de las palabras.

Otros ejemplos a discutir se muestran a continuación:



Analisis de sentimientos en Twitter

```python
import tweepy
from textblob import TextBlob

# Autenticación en Twitter API (Es necesario las credenciales, el ejemplo es genérico, se discutirá mas adelante)
consumer_key = 'TU_CONSUMER_KEY'
consumer_secret = 'TU_CONSUMER_SECRET'
access_token = 'TU_ACCESS_TOKEN'
access_token_secret = 'TU_ACCESS_TOKEN_SECRET'

auth = tweepy.OAuth1UserHandler(consumer_key, consumer_secret, access_token, access_token_secret)
api = tweepy.API(auth)

# Buscar tweets sobre un tema específico
tweets = api.search(q='Python', count=10)

# Analizar el sentimiento de cada tweet
for tweet in tweets:
    analysis = TextBlob(tweet.text)
    print(tweet.text)
    print('Sentimiento:', analysis.sentiment)
    print()
```

TIC

Clasificación de Texto con TensorFlow

```
#Este script utiliza TensorFlow y Keras para construir un modelo de red neuronal convolucional (CNN) para clasificar reseñas de películas como positivas o negativas.

import tensorflow as tf

from tensorflow.keras.datasets import imdb

from tensorflow.keras.preprocessing.sequence import pad_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, Conv1D, GlobalMaxPooling1D


# Cargar datos de IMDB

vocabulario = 10000

max_longitud = 200

(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=vocabulario)

X_train = pad_sequences(X_train, maxlen=max_longitud)

X_test = pad_sequences(X_test, maxlen=max_longitud)


# Construir modelo CNN

modelo = Sequential()
```

```
modelo.add(Embedding(vocabulario, 128,
input_length=max_longitud))
modelo.add(Conv1D(128, 5, activation='relu'))
modelo.add(GlobalMaxPooling1D())
modelo.add(Dense(1, activation='sigmoid'))

modelo.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# Entrenar modelo
modelo.fit(X_train, y_train, epochs=5, batch_size=32,
validation_data=(X_test, y_test))
```

Este anterior es un ejemplo muy genérico, se puede compilar para comprobar los parámetros vistos en la última parte de dicho scripts, épocas = 5, tamaño del lote = 32 etc, esto permite porcionar el entrenamiento de la clasificación de textos.