Tarea 14 - NLP

May 3, 2021

0.1 # Tarea 14

0.2 ## NLP

Por: Miguel Angel Soto Hernandez

0.3 Importaciones

[ ]:
```
import os.path
import nltk
from gensim import corpora
from gensim.models import LsiModel
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from gensim.models.coherencemodel import CoherenceModel
import matplotlib.pyplot as plt
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[ ]:
```
True
```

[ ]:
```
def cargar_datos(path, nombre_archivo):
    """
    Entrada : ruta y nombre_de_archivo
    Finalidad: cargar un archivo de texto
    Salida : lista de párrafos/documentos y título (las 100 palabras iniciales
             se consideran el título del documento)
    """
    lista_documentos = []
    titulos = []

    with open(os.path.join(path, nombre_archivo) ,"r") as fin:
        for line in fin.readlines():
            texto = line.strip()
            lista_documentos.append(texto)
```

```
    print(f'Numero total de documentos: {len(lista_documentos)}')
    titulos.append(texto[0:min(len(texto), 100)])

    return lista_documentos, titulos
```

[ ]:
```
path = '/content/drive/MyDrive/Colab Notebooks/CIC/1 semestre/NLP/'
nombre_archivo = 'articles4.txt'
lista_documentos, titulos = cargar_datos(path, nombre_archivo)
```

```
Numero total de documentos: 4551
```

[ ]:
```
def preprocesar_datos(doc_set):
    """
    Entrada: lista de documentos
    Propósito: preprocesar el texto (tokenizar, eliminar las palabras de␣
 ↪parada,
                y stemming)
    Salida: texto preprocesado
    """
    # initializar tokenizador regex
    tokenizador = RegexpTokenizer(r'\w+')

    # crear lista de stopwords del Ingles
    stopwords_ingles = list(stopwords.words('english'))

    # crear p_stemmer de la clase PorterStemmer
    p_stemmer = PorterStemmer()

    # lista de documentos tokenizados en el ciclo
    textos = []

    # ciclo a traves de la lista de documentos
    for i in doc_set:
        # limpiar y tokenizar los strings del documento
        raw = i.lower()
        tokens = tokenizador.tokenize(raw)

        # quitar las stopwords de los tokens
        tokens_sin_stopwords = [i for i in tokens if i not in stopwords_ingles]

        # stem tokens
        stem_tokens = [p_stemmer.stem(i) for i in tokens_sin_stopwords]

        # agregar tokens a la lista
        textos.append(stem_tokens)

    return textos
```

[ ]:
```
textos = preprocesar_datos(lista_documentos)
```

[ ]:
```
def preparar_corpus(doc_limpio):
    """
    Entrada: documento limpio
    Objetivo: crear un diccionario de términos de nuestro corpus y convertir
              la lista de documentos (corpus) en la matriz de términos del
              documento
    Salida : diccionario de términos y matriz de términos del documento
    """
    '''
    Creando el diccionario de términos de nuestros corpus, donde a cada término
    único se le asigna un índice. dictionary = corpora.Dictionary(doc_limpio)
    '''
    diccionario = corpora.Dictionary(doc_limpio)

    '''
    Convertir la lista de documentos (corpus) en una matriz de términos de
    documentos utilizando el diccionario preparado anteriormente.
    '''
    matriz_term_doc = [diccionario.doc2bow(doc) for doc in doc_limpio]

    # generar modelo LDA
    return diccionario, matriz_term_doc
```

[ ]:
```
diccionario, matriz_term_doc = preparar_corpus(textos)
```

[ ]:
```
def crear_modelo_gensim(doc_limpio, numero_temas, palabras):
    """
    Entrada: documento limpio, número de temas y número de palabras asociadas a
            cada tema con cada tema
    Objetivo: crear un modelo LSA con gensim
    Salida: devolver el modelo LSA
    """
    dictionary, doc_term_matrix = preparar_corpus(doc_limpio)

    # generar modelo LSA
    modelo_lsa = LsiModel(doc_term_matrix, num_topics=numero_temas,
                          id2word = dictionary)
    print(modelo_lsa.print_topics(num_topics=numero_temas, num_words=palabras))

    return modelo_lsa
```

[ ]:
```
numero_temas = 7
palabras = 10
modelo_lsa = crear_modelo_gensim(textos, numero_temas, palabras)
```

```
[(0, '0.361*"trump" + 0.272*"say" + 0.233*"said" + 0.166*"would" +
0.160*"clinton" + 0.140*"peopl" + 0.136*"one" + 0.126*"campaign" + 0.123*"year"
```

```
+ 0.110*"time"'), (1, '-0.389*"citi" + -0.370*"v" + -0.356*"h" + -0.355*"2016" +
-0.354*"2017" + -0.164*"unit" + -0.159*"west" + -0.157*"manchest" + -0.116*"apr"
+ -0.112*"dec"'), (2, '0.612*"trump" + 0.264*"clinton" + -0.261*"eu" +
-0.148*"say" + -0.137*"would" + 0.135*"donald" + -0.134*"leav" + -0.134*"uk" +
0.119*"republican" + -0.110*"cameron"'), (3, '-0.400*"min" + 0.261*"eu" +
-0.183*"goal" + -0.152*"ball" + -0.132*"play" + 0.128*"said" + 0.128*"say" +
-0.126*"leagu" + 0.122*"leav" + -0.122*"game"'), (4, '0.404*"bank" + -0.305*"eu"
+ -0.290*"min" + 0.189*"year" + -0.164*"leav" + -0.153*"cameron" +
0.143*"market" + 0.140*"rate" + -0.139*"vote" + -0.133*"say"'), (5,
'-0.310*"bank" + 0.307*"say" + 0.221*"peopl" + -0.203*"trump" + -0.166*"1" +
-0.164*"min" + -0.163*"0" + -0.152*"eu" + -0.152*"market" + 0.138*"like"'), (6,
'0.570*"say" + 0.237*"min" + -0.170*"vote" + 0.158*"govern" + -0.154*"poll" +
0.122*"tax" + 0.115*"bank" + 0.115*"statement" + 0.112*"budget" +
-0.108*"one"')]
```

[ ]:
```python
def computar_coherencia_de_valores(diccionario, matriz_term_doc, doc_limpio,
                                   parar, inicio=2, paso=3):
    """
    Entrada : diccionario : diccionario Gensim
              corpus : corpus Gensim
              textos : Lista de textos de entrada
              stop : Número máximo de temas
    propósito : Calcular la coherencia c_v para varios números de temas
    Salida : lista_modelo : Lista de modelos de temas LSA
              coherence_values : Valores de coherencia correspondientes al
              modelo LDA con el número respectivo de temas
    """
    coherencia_valores = []
    lista_modelo = []

    for num_topics in range(inicio, parar, paso):
        # generar modelo LSA
        modelo = LsiModel(matriz_term_doc, num_topics=numero_temas,
                          id2word = diccionario)
        lista_modelo.append(modelo)
        coherencemodel = CoherenceModel(model=modelo, texts=doc_limpio,
                                        dictionary=diccionario, coherence='c_v')
        coherencia_valores.append(coherencemodel.get_coherence())

    return lista_modelo, coherencia_valores
```

[ ]:
```python
def plotear_grafico(doc_limpio, inicio, parar, paso):
    diccionario, matriz_term_doc = preparar_corpus(doc_limpio)
    lista_modelo, coherencia_valores = \
      computar_coherencia_de_valores(diccionario, matriz_term_doc, doc_limpio,
                                     parar, inicio, paso)
    # mostrar grafico
    x = range(inicio, parar, paso)
```