Anexo 5

Proyecto 5: Creación de Words Embedding Nivel 2

Mg. Luis Felipe Bustamante Narváez

Word2Vec

Es un modelo que se utiliza para aprender representaciones vectoriales de palabras. Estas representaciones pueden capturar muchas propiedades lingüísticas de las palabras, como su significado semántico, gramatical y hasta contextual.

Para este segundo ejemplo, usaremos un texto más extenso llamado "Cien años de soledad" del escritor colombiano Gabriel García Márquez, que tiene alrededor de 139.318 palabras y 823735 caracteres.

Librerías

In [...
```
!pip install pypdf2
```

```
Requirement already satisfied: pypdf2 in c:\users\luis_\anaconda3\envs\notebook\lib\site-packages (3.0.1)
```

In [...
```
import string
from gensim.models import Word2Vec
import PyPDF2
```

Cargamos el documento

In [...
```
def extraer_texto_desde_pdf(ruta_archivo):
    with open(ruta_archivo, 'rb') as archivo:
        lector = PyPDF2.PdfReader(archivo)
        texto = ''
        for pagina in range(len(lector.pages)):
            texto += lector.pages[pagina].extract_text()
        return texto
```

In [...
```
documento = extraer_texto_desde_pdf('Entrenamiento_Word2Vec/cien_anos_de_soledad.pdf')
```

In [...
```
len(documento)
```

Out[...
```
823735
```

Procesamiento de datos

El objetivo del procesamiento es convertir el documento en una lista de frases, y cada frase en una lista de palabras, eliminando signos de puntuación y convirtiendo todo a minúsculas.

In [...
```
# Dividimos el documento en frases usando la coma como separador
frases = documento.split(',')
len(frases)
```

Out[...
```
8854
```

In [...
```
# Mostramos un ejemplo
frases[500]
```

Out[...
```
' las huestes de su padre tenían un aspecto de náufragos sin escapatoria'
```

In [...]
```
# Mostramos un ejemplo
frases[3000]
```

Out[...]
```
' sino porque lo único que me faltó fue darte de mamar.» Aureliano escapaba al alba y regresaba a la madrugada siguiente'
```

In [...]
```
# Limpiamos las frases
frases_limpias = []
for frase in frases:
    #Eliminamos la puntuación y dividimos por espacios
    tokens = frase.translate(str.maketrans('','',string.punctuation)).split()
    #print(tokens)  #para mostrar qué ha hecho hasta aquí
    #Convertimos a minúsculas
    tokens = [word.lower() for word in tokens]
    #print(tokens)  #para mostrar qué ha hecho hasta aquí
    if tokens:
        frases_limpias.append(tokens)
```

In [...]
```
# Mostramos los resultados
frases_limpias[500]
```

Out[...]
```
['las',
 'huestes',
 'de',
 'su',
 'padre',
 'tenían',
 'un',
 'aspecto',
 'de',
 'náufragos',
 'sin',
 'escapatoria']
```

Entrenamiento del modelo Word2Vec

In [...]
```
model = Word2Vec(sentences=frases_limpias,
                 vector_size=500,
                 window=5,
                 min_count=1,
                 workers=4)
```

Explicación:

- sentences: Es la lista de palabras que vamos a vectorizar
- vector_size: Es el tamaño de dimensiones que le daremos al vector
- window: Son la cantidad de palabras por encima y por debajo que le darán contexto
- min_count: La aparición mínima de una palabra para tenerla en cuenta en el entrenamiento
- workers: Cantidad de núcleo de procesador que vamos a invertir en el entrenamiento

Pruebas

In [...]
```
# Verificamos el vector para alguna palabra
vector = model.wv['padre']
vector
```

In [...]
```
# Mostramos las palabras cercanas
palabras_cercanas = model.wv.most_similar('buendía', topn=10)
palabras_cercanas
# Es probable que la similitud falle por tener tan pocas palabras en el texto
```

Out[…
```
[('segundo', 0.9980278611183167),
 ('josé', 0.9935610890388489),
 ('buendia', 0.9925903081893921),
 ('tío', 0.9853704571723938),
 ('aureliano', 0.9842246770858765),
 ('promovió', 0.9841890335083008),
 ('pequeño', 0.9830081462860107),
 ('participó', 0.9812502264976501),
 ('informaciones', 0.9800763726234436),
 ('arcadio', 0.9795325398445129)]
```

Guardar modelo

In […
```
model.save('Entrenamiento_Word2Vec/cien_anos_de_soledad.model')
```

Cargar el modelo

In […
```
modelo_cargado = Word2Vec.load('Entrenamiento_Word2Vec/cien_anos_de_soledad.model')
```

In […
```
# Probamos con el modelo caragado
palabras_cercanas2 = modelo_cargado.wv.most_similar('buendía', topn=10)
palabras_cercanas2
```

Out[…
```
[('segundo', 0.9980278611183167),
 ('josé', 0.9935610890388489),
 ('buendia', 0.9925903081893921),
 ('tío', 0.9853704571723938),
 ('aureliano', 0.9842246770858765),
 ('promovió', 0.9841890335083008),
 ('pequeño', 0.9830081462860107),
 ('participó', 0.9812502264976501),
 ('informaciones', 0.9800763726234436),
 ('arcadio', 0.9795325398445129)]
```

Guardar Embedido

Existen dos maneras, usando .txt sin binarios, y usando .bin con binarios.

In […
```
model.wv.save_word2vec_format('Entrenamiento_Word2Vec/cien_anos_de_soledad_emb.txt', binary=False)
model.wv.save_word2vec_format('Entrenamiento_Word2Vec/cien_anos_de_soledad_emb.bin', binary=True)
```

Cargar Embedidos

Si se carga el .txt, se usa sin binarios; si se carga el .bin, se usa con binarios

In […
```
from gensim.models import KeyedVectors
embedding_cargado_txt = KeyedVectors.load_word2vec_format(
    'Entrenamiento_Word2Vec/cien_anos_de_soledad_emb.txt', binary=False)
```

In […
```
embedding_cargado_bin = KeyedVectors.load_word2vec_format(
    'Entrenamiento_Word2Vec/cien_anos_de_soledad_emb.bin', binary=True)
```

In […
```
# Probamos
embedding_cargado_txt
```

Out[…
```
<gensim.models.keyedvectors.KeyedVectors at 0x1116c4d67b0>
```

In […
```
# Probamos
embedding_cargado_bin
```

Out[…
```
<gensim.models.keyedvectors.KeyedVectors at 0x1116c43a390>
```

Analogías

In [...
```
def analogics(v1, v2, v3):
    simil = embedding_cargado_bin.most_similar(positive=[v1,v3],
                                               negative=[v2]
                                              )
    print(f'{v1} es a {v2}, como {simil[0][0]} es a {v3}')
```

In [...
```
analogics('aureliano', 'buendía', 'iguarán')
```

```
aureliano es a buendía, como luz es a iguarán
```

Conclusiones

El texto Cien años de soledad, tiene 823735 caracteres, lo que implica una base de datos más grande para entrenar un modelo, pero sigue siendo pequeña para un modelo adecuado. De cierta forma, el modelo se ajusta en algunos casos puntuales, pero suele mostrar demasiadas stopwords en las similitudes, que tendríamos que manipular para mejorar la predicción de analogías. Veremos con un paquete de textos más grandes y variados, como se generaría la predicción.

Mg. Luis Felipe Bustamante Narváez