UNIVERSIDAD CENTROCCIDENTA

“LISANDRO ALVARADO”

DECANATO DE CIENCIAS YTECNOLOGÍA

Laboratorio unidad IV

Introducción a la Inteligencia Artificial

Sección 1

V-27.411.922, Castillo Gustavo

V-25.401.656, Colmenárez Luis

V-28.454.911, Rojas Gabriel

Profesora: María Perez

Barquisimeto, diciembre de 2024

Índice

Resumen

Este informe documenta el entrenamiento y evaluación de un modelo de aprendizaje automático basado en la técnica Doc2Vec de la biblioteca Gensim, utilizando tres conjuntos de datos: Wikipedia, boletines, y boletines de prueba. La metodología consistió en procesar los datos textuales mediante técnicas de limpieza como la eliminación de puntuación y palabras irrelevantes, para luego etiquetar y construir representaciones vectoriales que preserven el significado semántico de los documentos.

El modelo se entrenó en iteraciones ajustando parámetros como la tasa de aprendizaje. Con los datos de Wikipedia, se logró construir un modelo genérico para representar texto general. Posteriormente, los conjuntos de boletines y boletines de prueba fueron empleados para personalizar el modelo hacia el análisis de datos especializados, evaluando similitudes entre documentos.

Los resultados mostraron que el modelo puede identificar relaciones entre documentos con base en similitudes semánticas, lo que sugiere una representación adecuada de los datos. Las conclusiones destacan la eficacia del enfoque Doc2Vec para tareas de clasificación y agrupamiento de texto en dominios específicos, así como la importancia de datos limpios y variados en el entrenamiento para mejorar la generalización del modelo.

Introducción

En la era de la información, el análisis eficiente de datos textuales se ha convertido en una herramienta fundamental para diversas aplicaciones, como la búsqueda de información, la clasificación de documentos, y el análisis de contenido. Una de las técnicas más avanzadas en este campo es el modelo Doc2Vec, que permite representar documentos como vectores numéricos conservando relaciones semánticas.

Este informe detalla el proceso de entrenamiento y evaluación de un modelo Doc2Vec utilizando tres conjuntos de datos: definiciones generales extraídas de Wikipedia y textos especializados de boletines. El objetivo principal es explorar la capacidad del modelo para adaptarse a diferentes dominios y evaluar su desempeño al identificar relaciones semánticas en documentos previamente etiquetados.

Se describe la metodología utilizada para limpiar, procesar y etiquetar los datos, así como las configuraciones empleadas durante el entrenamiento. Finalmente, se presentan los resultados obtenidos en términos de similitudes documentales y las implicaciones prácticas del modelo para aplicaciones futuras en análisis de texto especializado.

Objetivos de la investigación

General:

Desarrollar y evaluar un modelo Doc2Vec para la representación semántica de documentos textuales en dominios generales y especializados.

Específicos:

- Implementar un modelo Doc2Vec utilizando datos generales de Wikipedia para explorar su capacidad de representar texto en contextos amplios.
- Entrenar y personalizar el modelo con datos específicos de boletines, evaluando su desempeño en tareas de agrupamiento y clasificación.
- Analizar la capacidad del modelo para identificar similitudes semánticas entre documentos en conjuntos de datos especializados y de prueba.
- Documentar la metodología y resultados obtenidos, resaltando las ventajas y limitaciones del enfoque Doc2Vec en el análisis de datos textuales.

Marco Teórico

1. Aprendizaje profundo (Deep Learning)

El aprendizaje profundo es una rama del aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas para procesar grandes volúmenes de datos. Este enfoque busca simular el funcionamiento del cerebro humano, permitiendo que las máquinas realicen tareas como reconocimiento de imágenes, procesamiento del lenguaje natural y predicción basada en patrones complejos.

El aprendizaje profundo se destaca por su capacidad para automatizar la extracción de características relevantes de los datos, eliminando en muchos casos la necesidad de ingeniería manual de características. Esto lo convierte en una herramienta poderosa para análisis de texto, entre otros campos.

2. Modelo de memoria distribuida (Paragraph Vector)

El modelo de memoria distribuida, también conocido como Paragraph Vector, es una extensión del modelo Word2Vec diseñado para representar textos completos (frases, párrafos o documentos) como vectores de densidad fija. Este modelo asigna una representación vectorial única a cada documento que captura su contexto semántico global, permitiendo tareas como clasificación, búsqueda y agrupamiento.

A diferencia de los enfoques basados únicamente en palabras, Paragraph Vector considera la relación entre palabras y documentos en su conjunto, lo que lo hace ideal para analizar conjuntos de datos textuales heterogéneos.

3. ¿Qué es dataset?

Un dataset es un conjunto estructurado de datos que se utiliza para entrenar, validar y probar modelos de aprendizaje automático. Los datasets suelen estar organizados en

registros y atributos, donde cada registro representa una instancia única y los atributos describen sus características.

3.1 Descripción de los conjuntos de datos del trabajo:

- **Wikipedia.txt**: Contiene definiciones generales separadas por saltos de línea. Este dataset es utilizado para entrenar el modelo en un dominio amplio y no especializado.
- **Boletines.csv**: Incluye textos especializados en el formato Idtexto;Texto;Supervisión. Sirve como fuente de entrenamiento para adaptar el modelo a un dominio específico.
- **Boletines Prueba.csv**: Posee la misma estructura que Boletines.csv, pero se utiliza para evaluar el desempeño del modelo en un conjunto independiente.

4. ¿Qué es Gensim y sus usos?

Gensim es una biblioteca de Python diseñada para el procesamiento y análisis de texto, enfocándose en la modelización de temas, similitud semántica y representación de texto mediante vectores. Entre sus usos destacan:

- Entrenamiento de modelos como Word2Vec, Doc2Vec y FastText.
- Generación de representaciones vectoriales de texto para clasificación y agrupamiento.
- Similitud semántica y recuperación de información.

5. Modelo pre entrenado Doc2Vec

¿Qué es y cómo funciona?

Doc2Vec es una extensión de Word2Vec que permite representar documentos enteros como vectores en un espacio semántico. Su principio se basa en asignar un vector único a cada documento, el cual es entrenado simultáneamente con los vectores de palabras para capturar el contexto global del texto.

Principios:

Usa un vector adicional para representar el documento junto con los vectores de palabras.

Durante el entrenamiento, se ajustan los pesos del vector del documento para maximizar la probabilidad de predecir palabras dentro del texto.

Predicción del modelo:

Una vez entrenado, Doc2Vec utiliza el vector del documento para encontrar textos similares, agrupar documentos o clasificar textos.

6. Diferencias entre Doc2Vec y Word2Vec

Aspecto	Word2Vec	Doc2Vec
Representación	Palabras individuales.	Documentos completos.
Aplicaciones	Análisis léxico.	Clasificación, agrupamiento y similitud de documentos.
Contexto semántico	Basado en palabras vecinas.	Considera el documento completo.
Vectores únicos	No.	Sí.

7. ¿Cómo entrenar nuestros propios vectores de palabras usando Gensim Doc2Vec?

7.1. Preparación de los datos:

- Limpiar el texto eliminando puntuación, caracteres especiales y palabras irrelevantes.
- Dividir el texto en frases o documentos etiquetados.

7.2. Etiquetado de documentos:

- Usar estructuras como TaggedDocument de Gensim para asignar identificadores únicos a cada documento.

7.3. Construcción del modelo:

- Inicializar un modelo Doc2Vec configurando parámetros como la dimensión del vector y la tasa de aprendizaje.

7.4. Entrenamiento:

- Utilizar el método train en múltiples iteraciones, ajustando la tasa de aprendizaje para evitar sobreajuste.

7.5. Evaluación:

- Utilizar el método most_similar para verificar la capacidad del modelo de identificar documentos relacionados.

7.6. Guardado:

- Guardar el modelo entrenado con save para reutilización y análisis posterior.

Resultados

1. Experimentación y entrenamiento del modelo gensim_doc2vec con data Wikipedia – Pasos realizados, entrenamiento con la data de Wikipedia, printscreen de corrida y salidas…).

1.1. Pasos realizados

1.1.1. Carga de datos: La experimentación se realizó con un archivo de texto grande (wikipedia.txt) que contiene información en formato conceptual. Dado que el tamaño del archivo (561 MB) puede dificultar su procesamiento, se adoptaron estrategias para manejar

```
8   # Leer archivo y tomar una muestra aleatoria de líneas
9   file_path = "wikipedia.txt"
10  sample_size = 5000
11  with open(file_path, "r", encoding="UTF-8") as file:
12      lines = file.readlines()  # Leer todas las líneas
13      sampled_lines = random.sample(lines, min(len(lines), sample_size))  # Seleccionar muestra aleatoria
```

el corpus de manera eficiente, utilizando una muestra representativa del archivo.

1.1.2. Preprocesamiento de datos:

I. Se divide el texto en oraciones: Usando una expresión regular que identifica los finales de las oraciones.

II. Divide las oraciones en palabras y se ignoran oraciones demasiado cortas.

III. Se crea una etiqueta única para cad a oración.

IV. Se Almacena cada oración procesada en una lista de documentos.

```python
# Preprocesar las líneas seleccionadas
sentenceEnders = re.compile(r'[.!?]\s+')
exclude = set(string.punctuation)
LabelDoc = namedtuple('LabelDoc', 'words tags')
all_docs = []
for idx, line in enumerate(sampled_lines):
    sentences = sentenceEnders.split(line)  # Dividir por signos de puntuación
    for sentence in sentences:
        sentence = ''.join(ch for ch in sentence if ch not in exclude)  # Quitar puntuación
        words = sentence.split()  # Dividir en palabras
        if len(words) < 3:  # Ignorar frases muy cortas
            continue
        tag = [f"SEN_{idx}"]  # Etiqueta única
        all_docs.append(LabelDoc(words, tag))
```

1.1.3. Entrenamiento del modelo Doc2Vec:

I. Se configura el modelo Doc2Vec con 50 dimensiones de vector, un mínimo de 2 ocurrencias de palabras, 10 épocas de entrenamiento y una tasa de aprendizaje que disminuye gradualmente.

```python
# Entrenar el modelo Doc2Vec
model = Doc2Vec(vector_size=50, min_count=2, epochs=10, alpha=0.025, min_alpha=0.025)
model.build_vocab(all_docs)

print("Entrenando el modelo...")
for epoch in range(10):
    print(f"Época {epoch+1}")
    model.train(all_docs, total_examples=model.corpus_count, epochs=model.epochs)
    model.alpha -= 0.002  # Reducir la tasa de aprendizaje
    model.min_alpha = model.alpha

# Guardar el modelo
model.save('wikipedia_sample.doc2vec')
```

II. Se construye el vocabulario del modelo a partir de los documentos procesados.

III. El modelo se entrena durante 10 épocas, ajustando los pesos con cada pasada y reduciendo la tasa de aprendizaje gradualmente.

IV. El modelo entrenado se guarda en un archivo para su uso futuro.

1.1.4. Evaluación del modelo:

I. Se selecciona un documento aleatorio de la lista all_docs y se imprime su contenido.

II. Se calcula la similitud entre el documento seleccionado y los otros documentos del modelo usando **most_similar**. Los 10 documentos más similares se recuperan y se imprimen junto con su nivel de similitud.

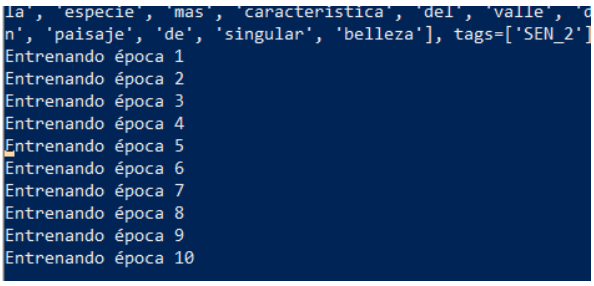
```
# Análisis de similitud
doc_id = np.random.randint(len(all_docs))  # Seleccionar un documento aleatorio
print(f"Documento objetivo (ID={doc_id}): {all_docs[doc_id].words}")

sims = model.dv.most_similar(f"SEN_{doc_id}", topn=10)  # Documentos más similares
print("\nDocumentos similares:")
for sim_id, similarity in sims:
    similar_idx = int(sim_id.replace("SEN_", ""))
    print(f"{sim_id} (similaridad: {similarity:.4f}): {all_docs[similar_idx].words}")
```

1.2 Resultado

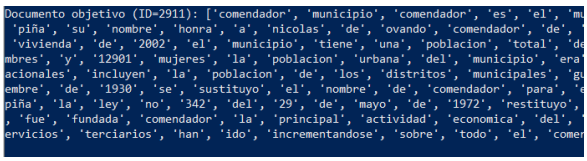
I. El código seleccionará una muestra aleatoria de hasta 5000 líneas del archivo wikipedia.txt. Si el archivo tiene menos de 5000 líneas, seleccionará todas las disponibles.

II. El modelo Doc2Vec se entrenará en 10 épocas con las oraciones preprocesadas. En cada época, el modelo ajustará sus parámetros para mejorar la representación de los documentos.

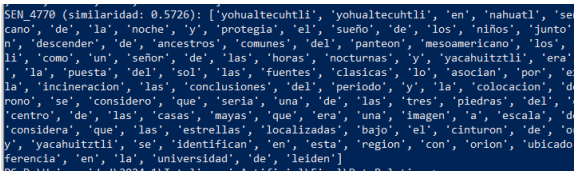
III. Durante el entrenamiento, la tasa de aprendizaje (alpha) disminuirá gradualmente en 0.002 después de cada época.



IV. Se muestra el contenido del documento seleccionado aleatoriamente como punto de partida del análisis.



V. Se calcularán los 10 documentos más similares según el modelo, como el que se presenta a continuación:



2. Entrenando al modelo gensim_doc2vec con conjunto de datos denominado boletines (paso realizados, entrenamiento con la data boletines y printscreen de corrida…)

2.1. Pasos realizados

2.1.1. Carga de datos:

I. Se utilizó el archivo boletines.csv, compuesto por los campos Idtexto, Texto y Supervision, separados por punto y coma (;).

II. Se concatenó el contenido del campo Texto en un solo bloque para su procesamiento posterior**.**

```python
# Leer el archivo CSV
with open("boletines.csv", newline='', encoding='UTF-8', errors='ignore') as csvfile:
    reader = csv.reader(csvfile)
    data = ''
    count = 0
    for row in reader:
        count += 1
        if count > 55:
            break
        # Verificar que la fila tenga al menos dos columnas
        if len(row) > 1:
            data += row[1]  # Concatenar solo la columna de texto
        else:
            print(f"Fila {count} no tiene suficientes columnas: {row}")
```

2.1.2. Preprocesamiento de datos:

I. Se dividieron los textos en oraciones utilizando una expresión regular.

II. Se eliminaron caracteres de puntuación para limpiar los datos.

```python
# Configurar una expresión regular para dividir el párrafo en oraciones.
sentenceEnders = re.compile('[.?!]')
data_list = sentenceEnders.split(data)

# Crear un namedtuple con words y tags
LabelDoc = namedtuple('LabelDoc', 'words tags')
exclude = set(string.punctuation)
all_docs = []
count = 0
for sen in data_list:
    word_list = sen.split()
    if len(word_list) < 3:
        continue
    tag = ['SEN_' + str(count)]
    count += 1
    sen = ''.join(ch for ch in sen if ch not in exclude)
    all_docs.append(LabelDoc(sen.split(), tag))

print(all_docs[0:10])
```

III. Se descartaron oraciones con menos de tres palabras, ya que aportan poca información semántica.

IV. Cada oración se etiquetó de manera única (SENT_0, SENT_1, entre otros).

2.1.3. Entrenamiento del modelo Doc2Vec:

I. Se inicializa un modelo Doc2Vec con parámetros específicos (tasa de aprendizaje fija).

```python
# Entrenar el modelo Doc2Vec
model = Doc2Vec(alpha=0.025, min_alpha=0.025)  # usar tasa de aprendizaje fija
model.build_vocab(all_docs)
for epoch in range(10):
    model.train(all_docs, total_examples=model.corpus_count, epochs=model.epochs)
    model.alpha -= 0.002  # disminuir la tasa de aprendizaje
    model.min_alpha = model.alpha  # fijar la tasa de aprendizaje, sin decaimiento

model.save('boletinesModel.doc2vec')
```

II. Se construye un vocabulario a partir de los documentos.

III. El modelo se entrena en 10 iteraciones, ajustando los pesos para optimizar los vectores de documentos.

IV. Finalmente, se guarda el modelo entrenado para usarlo posteriormente en tareas como inferencia o similitud de documentos.

2.1.4. Evaluación del modelo:

I. Se selecciona un documento objetivo aleatorio y se imprime

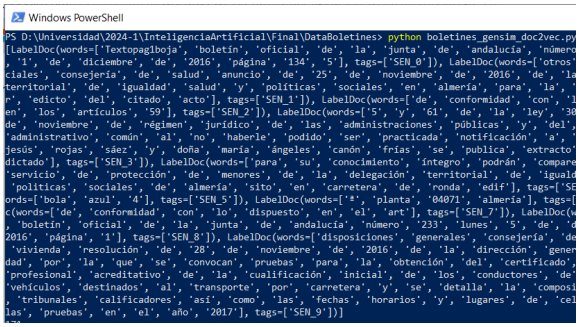
II. Se utiliza el modelo Doc2Vec para encontrar los documentos más cercanos en el espacio vectorial.

```python
# Obtener un documento aleatorio
doc_id = np.random.randint(model.dv.vectors.shape[0])
print(doc_id)

sims = model.dv.most_similar(doc_id, topn=model.dv.vectors.shape[0])
print('TARGET', all_docs[doc_id].words)

count = 0
for i in sims:
    if count > 8:
        break
    pid = int(i[0].replace("SEN_", ""))
    print(i[0], ": ", all_docs[pid].words)
    count += 1
```

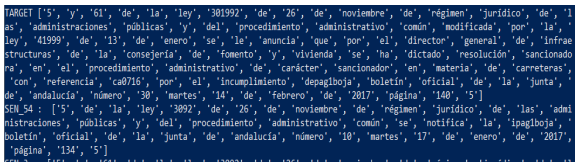
III. Se imprimen los resultados: Mostrando el contenido del documento objetivo y hasta 8 documentos similares, junto con sus identificadores.

2.2. Resultados:

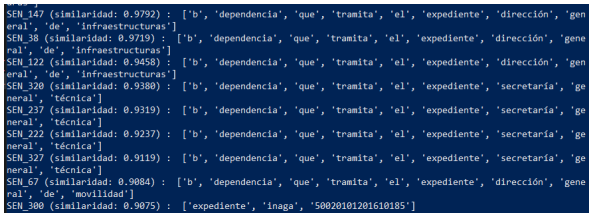
I. **LabelDoc(words)**: Representa el contenido del texto del documento después de separar palabras y eliminar signos de puntuación.



II. Muestra el contenido del documento seleccionado aleatoriamente como punto de partida del análisis.



III. Se muestra el contenido similar: Representado como una lista de palabras preprocesadas.



SEN: Es el identificador (tag) del documento similar. Los identificadores se asignan al preprocesar el corpus (por ejemplo, SEN_147, SEN_38, SEN_122, entre otros).

Similaridad: Representa el porcentaje de similitud que tiene con el objetivo, el SEN 147 posee una similaridad de 0.9792, lo cúal es cercano y superior al siguiente SEN_38, que posee un similaridad de 0.9719.

Relaciones semánticas captadas: El modelo ha detectado relaciones significativas entre conceptos de física relacionados con la gravedad.

Jerarquía de similitud: Los documentos con mayor contenido compartido (términos y contexto) reciben puntajes de similitud más altos.

3. Entrenando al modelo gensim_doc2vec con conjunto de datos denominado boletines de prueba (paso realizados, entrenamiento con boletines de prueba y printscreen de corrida…)

3.1. Pasos realizados

3.1.1. Carga de datos:

Se utilizó el archivo **boletinesprueba.csv**, que posee los campos Idtexto, Texto y Supervisión, pero a diferencia de **boletines.csv** posee los valores separados por coma (,). Por lo que cambiamos el delimitador.

Se repiten los mismos pasos que con boletines.csv: **2.1.2. Preprocesamiento de datos**, **2.1.3. Entrenamiento del modelo Doc2Vec**, **2.1.4. Evaluación del modelo.**

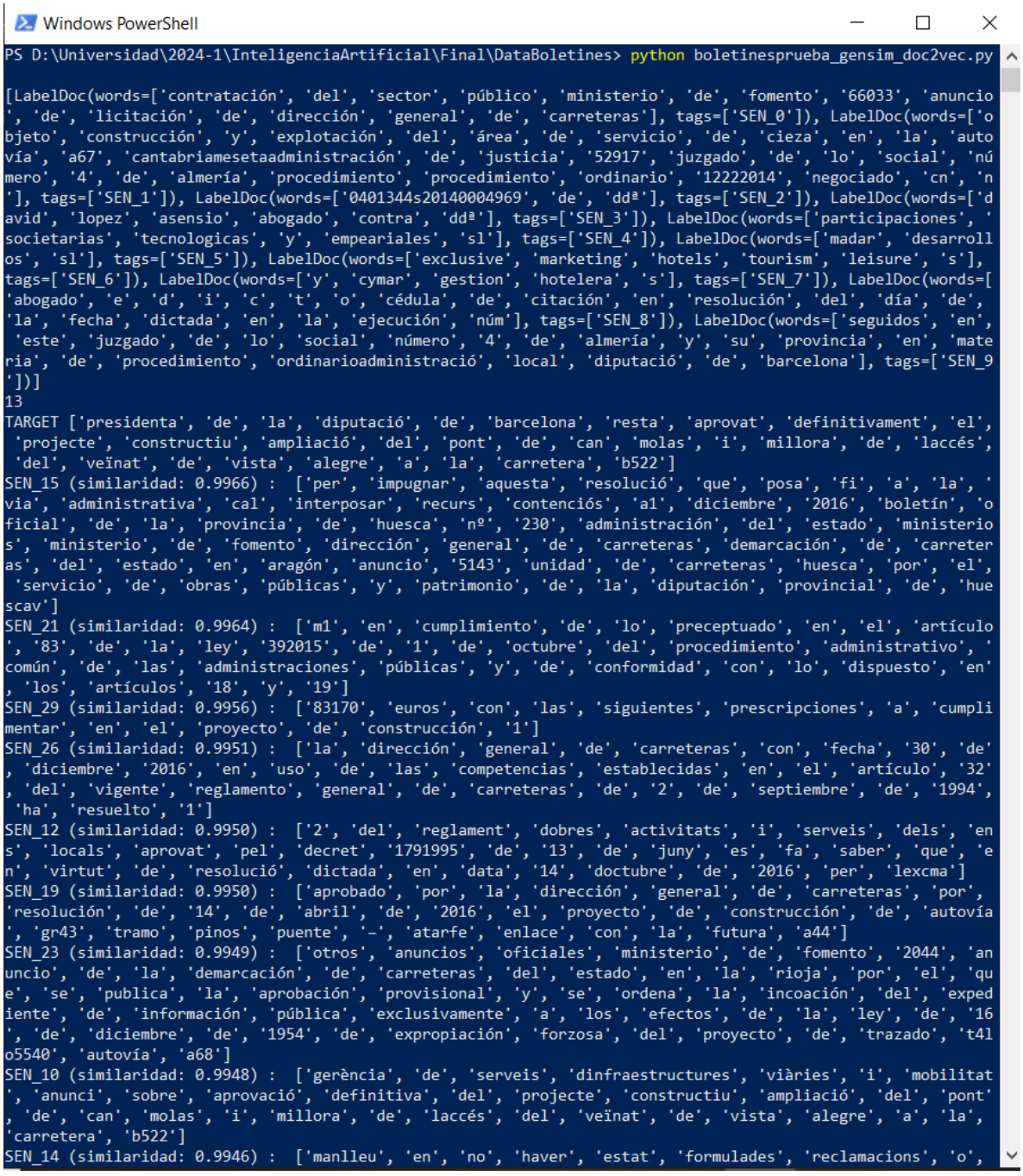
3.2. Resultados

- Se selecciona un documento aleatorio del corpus (TARGET) para compararlo con el resto. (Este documento está representado como una lista de palabras procesadas (sin puntuación)
- El modelo Doc2Vec busca los documentos más similares al objetivo en el espacio semántico. La similitud se mide como un valor entre -1 (completamente opuesto) y 1 (idéntico).
- Los resultados incluyen el identificador del documento (SEN_X), su contenido preprocesado y el puntaje de similitud.

Interpretación de los Resultados:

- Los documentos con puntajes de similitud más altos comparten términos clave o temas relacionados con el documento objetivo.

- Los documentos menos similares también pueden estar relacionados de manera más vaga, pero tienen menos coincidencias directas en palabras o contexto.



Conclusiones

Aplicabilidad de Doc2Vec: El modelo Doc2Vec, basado en Gensim, demuestra ser una herramienta efectiva para representar documentos como vectores densos, lo que facilita tareas como la clasificación, búsqueda y agrupamiento de textos. Su capacidad para capturar el contexto semántico global de los textos lo hace especialmente útil para conjuntos de datos especializados, como los boletines analizados.

Versatilidad del entrenamiento: El uso de datasets variados, como Wikipedia.txt y los archivos de boletines, evidencia la flexibilidad de Doc2Vec para adaptarse a diferentes dominios y contextos de trabajo. Entrenar el modelo con datos específicos mejora significativamente su capacidad para interpretar y analizar textos similares.

Importancia de la preprocesamiento: La calidad de los resultados obtenidos depende en gran medida del preprocesamiento de los datos. Pasos como la limpieza de texto, eliminación de caracteres irrelevantes y segmentación adecuada de frases son críticos para garantizar un entrenamiento efectivo del modelo.

Gensim como herramienta clave: Gensim se posiciona como una biblioteca robusta y accesible para el procesamiento de lenguaje natural, ofreciendo una implementación eficiente de Doc2Vec que permite a los investigadores entrenar modelos personalizados con relativa facilidad.

Perspectivas futuras: El enfoque empleado puede extenderse a otros conjuntos de datos y adaptarse para mejorar la comprensión semántica en diversas áreas, como análisis de sentimientos, recuperación de información y generación de resúmenes automáticos.

Referencias Bibliográficas

Le, Q., & Mikolov, T. (**2014**). Representaciones Distribuidas de Frases y Documentos.

Goldberg, Y., & Levy, O. (**2014**). Word2Vec Explicado: Derivando el Método de Embedding de Palabras con Muestreo Negativo de Mikolov.

Referencias Web

https://radimrehurek.com/gensim/: Radim Řehůřek. (2018). Gensim: Modelado de Temas para Humanos.

https://radimrehurek.com/gensim/models/doc2vec.html: Documentación de Doc2Vec en Gensim.

https://machinelearningmastery.com: Mastery en Aprendizaje Automático. (2021). Cómo Desarrollar un Modelo de Vector de Párrafo.

https://es.wikipedia.org/wiki/Aprendizaje_profundo: Colaboradores de Wikipedia. (2023). Aprendizaje Profundo.

https://towardsdatascience.com: Hacia la Ciencia de Datos. (2022). Doc2Vec y Word2Vec: Diferencias y Aplicaciones.