Trabajo Práctico 2

TUIA - Procesamiento del Lenguaje Natural

Celi, María

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Universidad Nacional de Rosario

Av. Pellegrini 250 - 2000 Rosario – Argentina.

Introducción

El siguiente trabajo es en el marco de la materia Procesamiento del lenguaje Natural de la Tecnicatura en Inteligencia Artificial (TUIA). En él se explicará el desarrollo de los 6 ejercicios propuestos por la cátedra.

Ejercicio 1

Para realizar el ejercicio 1 se descargaron las carpetas ‘información’, ‘relaciones’, ‘estadísticas’, se creó un nuevo archivo .ipbyn donde se desarrolló el trabajo práctico y el mismo se subió en el siguiente repositorio:

Ejercicio 2

En este ejercicio se debía realizar un análisis de similitud de texto ingresando varias frases a buscar semánticamente. Para ello comencé dividiendo el texto manual.txt con expresiones regulares por ‘.’, ‘?’ o ‘!’ y eliminé fragmentos cortos de texto que podrían generar ruido en el análisis:

```
sentences = re.split(r'(?<=[.!?]) +', text)
sentences = [s.strip() for s in sentences if len(s.strip()) > 20]
```

Luego opté por utilizar el modelo SBERT porque entiendo que está entrenado específicamente para tareas sobre análisis semántico, que es lo que necesitamos en este caso. También es ligero y con él podemos generar los embeddings significativos para el posterior análisis de similitud.

Después se realizó la vectorización del texto, como pedía la consigna, se generaron representaciones vectoriales del texto y de las consultas para poder comparar similitudes.

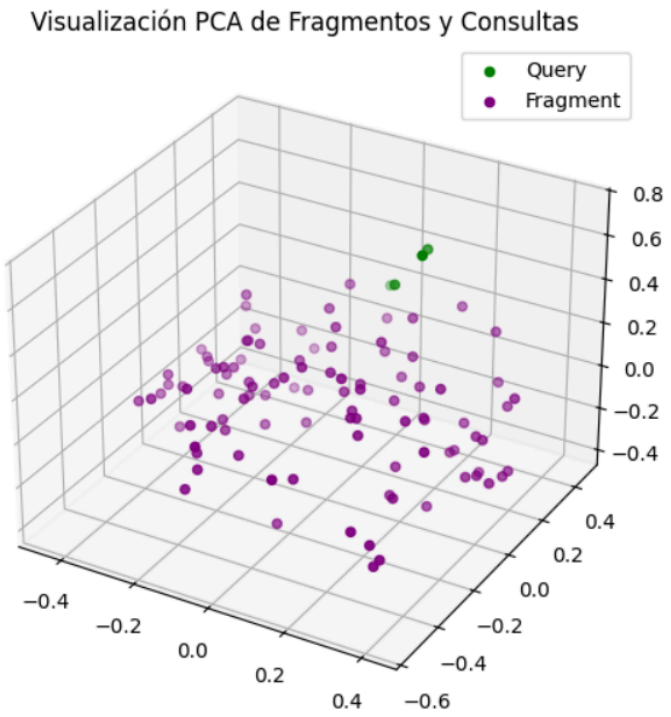
Por último, realizamos el análisis comparativo por las tres métricas elegidas Coseno, Manhattan y Euclídea. Elegí estas métricas porque se quiso trabajar directamente con embeddings y las métricas vistas en clases tal como Jaccard, Levenshtein o Dice, trabajan con el texto puro.

- **Similitud del coseno**: mide el ángulo entre vectores, sin importar su magnitud.
- **Distancia euclídea**: mide la distancia recta entre puntos en el espacio vectorial.
- **Distancia Manhattan**: suma de las diferencias absolutas entre coordenadas.

Por último hice un top 3 de las mejores coincidencias de cada una de las métricas y la visualización 3D con PCA, la cual me ayudó a ver cómo variaba la posición con respecto de los demás fragmentos de las querys en el espacio a medida que agregaba o quitaba más oraciones a comparar semánticamente.

En conclusión, quien resultó ser más consistente en términos semánticos fue la similitud por el coseno con fragmentos que más o menos se acercan a la respuesta de las consultas o al menos tienen que ver con la temática de la pregunta. La distancia euclídea dio resultados muy similares a la del coseno, pero esta métrica podría ser menos confiable con un texto más complejos porque como mide la distancia lineal entre dos puntos, probablemente con más ruido en el set de datos sería más sensible. Y por último, la distancia de Manhattan mostró resultados erráticos o muy variantes de repente, en algunos casos se alinea con las otras dos métricas y luego empieza a incluir nombres y fragmentos de textos no relacionados al tema.

	Query	Top Cosine	Top Euclidean	Top Manhattan
0	Cómo se construyen edificios	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., After a resource has been placed, players have until the next resource is named to construct any buildings., A resource can only be used to construct 1 building — in other words, every building requires all of its resources.\nBuildings do not have to be constructed as soon as its resources are in place.]	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., After a resource has been placed, players have until the next resource is named to construct any buildings., A resource can only be used to construct 1 building — in other words, every building requires all of its resources.\nBuildings do not have to be constructed as soon as its resources are in place.]	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., After a resource has been placed, players have until the next resource is named to construct any buildings., peter mcpherson\n§elcome!\nYou are the mayor of a tiny town in the forest, where the smaller creatures of the woods have created a civilization hidden away from predators.]
1	Qué pasa cuando termina la partida	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., Play continues until the end conditions of the standard game are met., (You no longer take turns naming resources as the \nMaster Builder.)\nNote: A player may continue to place resources and name resources as the Master \nBuilder as long as they have space for the resources in their town, even if it will not be possible for them to construct any buildings.\nIf only 1 player is left, that player can continue to name resources as the sole Master \nBuilder until their town is completed.\nGame End\nWhen all players' towns are completed, the game ends immediately.]	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., Play continues until the end conditions of the standard game are met., (You no longer take turns naming resources as the \nMaster Builder.)\nNote: A player may continue to place resources and name resources as the Master \nBuilder as long as they have space for the resources in their town, even if it will not be possible for them to construct any buildings.\nIf only 1 player is left, that player can continue to name resources as the sole Master \nBuilder until their town is completed.\nGame End\nWhen all players' towns are completed, the game ends immediately.]	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., Play continues until the end conditions of the standard game are met., peter mcpherson\n§elcome!\nYou are the mayor of a tiny town in the forest, where the smaller creatures of the woods have created a civilization hidden away from predators.]
2	Qué significa cada recurso	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., building card anatomy\nA.\tCard Title\nB.\tBuilding Type\nC.\tArt\nD.\tResource Build Pattern\nE.\tBuilding Ability/Scoring\n10\n1 1 b a c d e\nRound Overview\n1.\t The Master Builder names a type of resource., 4.\t Sort the remaining Building cards into separate piles by the symbols on the back.]	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., building card anatomy\nA.\tCard Title\nB.\tBuilding Type\nC.\tArt\nD.\tResource Build Pattern\nE.\tBuilding Ability/Scoring\n10\n1 1 b a c d e\nRound Overview\n1.\t The Master Builder names a type of resource., 4.\t Sort the remaining Building cards into separate piles by the symbols on the back.]	[555 N El Camino Real #A393, \nSan Clemente, CA 92672, USA\nAll rights reserved., building card anatomy\nA.\tCard Title\nB.\tBuilding Type\nC.\tArt\nD.\tResource Build Pattern\nE.\tBuilding Ability/Scoring\n10\n1 1 b a c d e\nRound Overview\n1.\t The Master Builder names a type of resource., peter mcpherson\n§elcome!\nYou are the mayor of a tiny town in the forest, where the smaller creatures of the woods have created a civilization hidden away from predators.]
3	Cuántas personas pueden jugar	[Magnus, \nBrian Neugebauer, Sam LaFleche, Kevin \nPiala, Jed Moody, Martin Glazer, Emily \nGarrick, Jon "Ouch!" Luke, and all of the others who played this game., 6.\t Draw 1 card from each pile and place them face-up next to the Cottage card, so they are visible to all players., Fountain\nFountains cannot score more than \n.]	[Magnus, \nBrian Neugebauer, Sam LaFleche, Kevin \nPiala, Jed Moody, Martin Glazer, Emily \nGarrick, Jon "Ouch!" Luke, and all of the others who played this game., 6.\t Draw 1 card from each pile and place them face-up next to the Cottage card, so they are visible to all players., Fountain\nFountains cannot score more than \n.]	[Magnus, \nBrian Neugebauer, Sam LaFleche, Kevin \nPiala, Jed Moody, Martin Glazer, Emily \nGarrick, Jon "Ouch!" Luke, and all of the others who played this game., 6.\t Draw 1 card from each pile and place them face-up next to the Cottage card, so they are visible to all players., 11.\tGive each player 1 wooden monument (\n), which they should keep next to their Monument card.]



Ejercicio 3

En el ejercicio 3 se pedía recoger un texto extenso extraído, dividirlo en fragmentos, realizar extracciones de sustantivos (POS) y categorizar esos sustantivos (NER) y también realizar una búsqueda de similitud filtrando por sustantivos.

Lo primero que hice fue elegir el texto, elegí el mismo que en el ejercicio 2 por una cuestión de conveniencia. Luego, elegí usar el modelo en_core_web_sm de Spacy ya que como se trata de un texto en inglés, debería funcionar bien. Las oraciones se extrajeron con doc.sents, que devuelve un span correspondiente a la oración detectada por el modelo.

```
# cargar modelo de spacy
nlp = spacy.load("en_core_web_sm")

# cargamos el texto a analizar
with open("../datos/informacion/manual.txt", "r", encoding="utf-8") as f:
    texto = f.read()

# separamos las oraciones por spacy
doc = nlp(texto)
oraciones = [sent.text.strip() for sent in doc.sents if len(sent.text.strip()) > 20]
```

Se identificaron los sustantivos de las oraciones con token.pos == “NOUN” para buscar los sustantivos más representativos de cada fragmento.

```
# extraer sustantivos y entidades por oración
sustantivos_por_oracion = []
entidades_por_oracion = []

for sent in oraciones:
    doc_sent = nlp(sent)
    sustantivos = [token.text for token in doc_sent if token.pos_ == "NOUN"]
    entidades = [(ent.text, ent.label_) for ent in doc_sent.ents]
    sustantivos_por_oracion.append(sustantivos)
    entidades_por_oracion.append(entidades)
```

Se hizo la clasificación de entidades ent.label_ cupon NER y se conservaron todas ellas porque ayudan al enriquecimiento semántico del texto.

```
# extraer sustantivos y entidades por oración
sustantivos_por_oracion = []
entidades_por_oracion = []

for sent in oraciones:
    doc_sent = nlp(sent)
    sustantivos = [token.text for token in doc_sent if token.pos_ == "NOUN"]
    entidades = [(ent.text, ent.label_) for ent in doc_sent.ents]
    sustantivos_por_oracion.append(sustantivos)
    entidades_por_oracion.append(entidades)
```

Para calculara la similitud por coseno se hizo la vectorización con TF-IDF sobre los sustantivos extraídos (embeddings). Se eligió la oración con índice 0 para comparar. La distancia de Jaccard se hizo sobre conjuntos de palabras y la de Levenshtein sobre cadenas completas.

```
# vectorizar con TF-IDF
sustantivos_texto = [" ".join(s) for s in sustantivos_filtrados]
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(sustantivos_texto)

idx_consulta = 0
consulta_vector = tfidf_matrix[idx_consulta]
consulta_texto = sustantivos_texto[idx_consulta]
```

Comparación de resultados: Las oraciones encontradas por el coseno comparten un vocabulario muy alineados en las 3. Por otro lado, Jaccard también comparte varias palabras clave entre todas sus oraciones, pero no tienen un orden o estructura similar al igual que lo tenía la métrica del coseno (ignora estructura y gramática). Y por último, en Levenshtein tenemos muchas palabras en común pero es más literal porque están en el mismo orden parece ser muy sensible al orden y a pequeñas variaciones.

Por lo tanto la métrica más adecuada parece ser la del coseno, teniendo en cuenta que para su aplicación, antes debemos vectorizar.

Ejercicio 4

El ejercicio 4 pedía mediante detección de idioma, separar los archivos en distintos lenguajes y guardar esa información en un dataframe. Para ello, primero instalé la librería 'langdetect' que emplea un clasificador Naive Bayes con n-gramas de caracteres para distinguir entre múltiples idiomas (Language Detection).

Se continuó importando las librerías necesarias, os para acceder al sistema de archivos, pandas para el dataframe y langdetect.detect() función que va a detectar el idioma. DetectorFactory.seed = 0 garantiza que el algoritmo arroje siempre los mismos resultados si se repite el análisis con los mismos datos.

```
import os
import pandas as pd
from langdetect import detect, DetectorFactory
DetectorFactory.seed = 0
```

Defino la ruta hacia la carpeta información y hago una lista vacía donde se acumularán los datos.

```
carpeta = '../datos/informacion'


datos = []
```

En el siguiente código, primero se recorren todos los archivos en la carpeta de información y se filtran solo aquellos que tienen extensión .txt. Se abren con codificación utf-8 para que sea compatible con carácteres especiales.

```
for archivo in os.listdir(carpeta):
    if archivo.endswith('.txt'):
        ruta_archivo = os.path.join(carpeta, archivo)
        with open(ruta_archivo, 'r', encoding='utf-8') as f:
            contenido = f.read()
            try:
                idioma = detect(contenido)
            except:
                idioma = 'desconocido'
            datos.append({'archivo': archivo, 'idioma': idioma})
```

Se lee todo el contenido y se pasa al detector de idioma. Si la detección falla (archivo vacío, error interno, etc.), se asigna el valor ‘desconocido’. Se agrega cada archivo y su idioma detectado a la lista datos.

Finalmente se crea y se muestra el dataframe.

```
df_idiomas = pd.DataFrame(datos)
print(df_idiomas)
```

```
              archivo idioma
0          manual.txt     en
1      review_bgg.txt     es
2  review_externa.txt     es
3          video1.txt     nl
4          video2.txt     en
5          video3.txt     en
6          video4.txt     es
```

Fue posible detectar textos en inglés, español y neerlandés de forma precisa.

Ejercicio 5

El ejercicio 5 dice lo siguiente: En el caso de las reseñas realizadas por usuarios, utiliza análisis de sentimientos con modelos preentrenados y guarda la clasificación predecida de cada reseña. Luego, crea un sistema de búsquedas por similitud semántica y que permita filtrar por sentimiento para obtener.

Para resolver este ejercicio primero se utilizó el modelo basado en BERT ‘nlptown/bert-base-multilingual-uncased-sentiment’ que clasifica los textos en escalas de estrellas, lo que facilita el análisis.

```
#!pip install transformers

from transformers import BertTokenizer, BertForSequenceClassification, pipeline
import pandas as pd

# cargar modelo preentrenado
model_name = "nlptown/bert-base-multilingual-uncased-sentiment"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)

# pipeline de análisis de sentimiento
nlp = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)
```

Loading...

Luego cargamos las reseñas a analizar y se lo aplicamos. Omitimos líneas demasiado cortas para que el modelo pueda funcionar con más precisión y obtenemos la predicción y además una medida de confianza.

```python
# cargar reseñas
with open("../datos/informacion/review_bgg.txt", "r", encoding="utf-8") as f:
    lineas = [linea.strip() for linea in f if len(linea.strip()) > 15]

# análisis de sentimiento
resultados = []
for linea in lineas:
    resultado = nlp(linea)[0]
    resultados.append({
        "texto": linea,
        "sentimiento_crudo": resultado["label"],
        "confianza": resultado["score"]
    })
```

En la siguiente función ‘clasificar’ lo que se hace es agrupar los valores del modelo en las tres clases más utilizadas para análisis de sentimiento.

```python
# campo de sentimiento a 'positivo', 'neutral', 'negativo'
def clasificar(label):
    estrellas = int(label[0])
    if estrellas <= 2:
        return "negativo"
    elif estrellas == 3:
        return "neutral"
    else:
        return "positivo"

df["sentimiento"] = df["sentimiento_crudo"].apply(clasificar)
display(df.head())
```

Pasamos a la parte de búsqueda semántica, donde utilizamos el modelo ‘msmarco-MiniLM-L-6-v3’ que es un corpus de recuperación de información a gran escala que fue creado basándose en consultas de búsqueda reales de usuarios utilizando el motor de búsqueda Microsoft Bing. Los modelos proporcionados pueden ser utilizados para búsqueda semántica, es decir, dado unas palabras clave

/ una frase de búsqueda / una pregunta, el modelo encontrará pasajes que son relevantes para la consulta de búsqueda.

Luego se hizo la vectotización y filtrado de reseñas y consulta. Primero filtramos la reseña por tipo de sentimiento, vectorizados y reducimos el espacio de búsqueda para mejorar la precisión del resultado:

```
# consulta y filtro
consulta = "me parece un juego muy aburrido"
sentimiento_deseado = "negativo"

#filtrar por sentimiento
df_filtrado = df[df["sentimiento"] == sentimiento_deseado].reset_index(drop=True)

# embeddings del corpus filtrado
corpus_embeddings = list(df_filtrado["embedding"])

# embedding de la consulta
embedding_consulta = modelo.encode(consulta)

embedding_consulta = modelo.encode(consulta, convert_to_tensor=True)
corpus_embeddings = modelo.encode(df_filtrado["texto"].tolist(), convert_to_tensor=True)
```

Por último realizamos la búsqueda semántica para obtener las oraciones más parecidas semánticamente a la consulta:

```
# calcular similitud
resultados = util.semantic_search(embedding_consulta, corpus_embeddings, top_k=3)[0]

# resultados
for i, r in enumerate(resultados):
    idx = r["corpus_id"]
    print(f"Resultado {i+1}:")
    print(f"Similitud: {r['score']:.4f}")
    print(f"Texto: {df_filtrado.iloc[idx]['texto']}")
    print()
```

Los resultados son bastante buenos, mi consulta fue “me parece un juego tan aburrido”, las consultas no dicen literalmente la palabra ‘aburrido’ pero la forma de expresarse de las oraciones puede ser interpretable como tal porque muestran sentimientos como cansancio (se pregunta cuándo se termina el juego) , insatisfacción o apatía. Se me ocurre como mejora hacer un programa que busque por todos los sentimientos y no solo por uno.

Ejercicio 6

El ejercicio 6 pedia crear un set de datos de consultas (más de 300 preguntas en total) y categorizarlas entre la fuente de datos que pueda llegar a responder esa pregunta entre estadísticas, información y relaciones.

Para ello, mediante un prompt con preguntas de ejemplo ya clasificadas, pedí a ChatGPT que me cree un set de 300 preguntas también ya clasificadas. Cargué las preguntas en un csv y cargué el archivo. También se instalaron y descargaron las librerías necesarias para vectorizar, dividir el dataset, los modelos que vamos a comparar y las métricas para hacerlo.

```
# LIBRERÍAS
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# cargamos preguntas
df = pd.read_csv("preguntas_clasificadas.csv")
```

Para la división; 20% del dataset se reserva para evaluar el rendimiento real del modelo:

```
# división del dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

Procedemos con la vectorización:

```
# vectorización
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Primero probamos el modelo multinomial Naive Bayes el cual es ligero y eficiente para texto categorizado y luego lo vamos a comparar con el modelo de regresión logística porque es uno de los clasificadores lineales más sólidos:

```
# modelo multinomial Naive Bayes
modelo_nb = MultinomialNB()
modelo_nb.fit(X_train_tfidf, y_train)
y_pred_nb = modelo_nb.predict(X_test_tfidf)

# modelo logistic regression
modelo_log = LogisticRegression(max_iter=1000)
modelo_log.fit(X_train_tfidf, y_train)
y_pred_log = modelo_log.predict(X_test_tfidf)
```

Se evaluaron accuracy, matriz de confusión y clasificación por clase. En Naive Bayes el accuracy fue de 0.986, el F1-score promedio ponderado de 0.99 y tuvo un error mínimo en la clase "informacion". En cambio, Logistic Regression tuvo un accuracy de 1 y una clasificación perfecta sin errores.

Conclusión final:

A lo largo de este trabajo se exploraron diferentes técnicas y etapas del Procesamiento del Lenguaje Natural aplicadas a la extracción previa, por parte de nuestros compañeros, de información sobre juegos de mesa. Cada ejercicio incursionó en diferentes aspectos vistos en clase para el análisis de textos, desde la similitud semántica, el análisis de sentimientos en un texto, la identificación de su idioma y la clasificación supervisada de preguntas. Cada aspecto fue clave para la

consolidación de las habilidades fundamentales para el NLP. Considero tener muchísimas áreas de mejora, si bien el resultado de los ejercicios creo que ha sido decente, en el proceso de la obtención del mismo se probaron muchas maneras de resolución. Más que nada en el ejercicio 3, donde en un primer momento no entendía bien la consigna, luego comencé a analizar el texto con un modelo entrenado en inglés con un texto en español y no clasificaba absolutamente ninguna palabra bien, luego lo intenté hacer con nltk, lo cual fue un poco engorroso y por último, volviendo sobre la teoría, consideré utilizar el modelo de spaCy entrenado en inglés con un texto en inglés y funcionó medianamente bien. Por lo tanto, este tipo de trabajos son un gran proceso de aprendizaje que fortalece el pensamiento crítico sobre, entre tantas tecnologías y herramientas presentadas, cuáles son las más coherentes dependiendo del contexto.