Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 5

Análisis de Sentimientos

Procesamiento de Lenguaje | Natural Language Processing

7CM3

Profesor: Joel Omar Juárez Gambino

Integrantes:

Benítez Anduaga Adrián

Reveles González Ángel Andrés

Martínez Segura Uriel Gerardo

Soto Avalos Edgar

Introducción

En esta práctica, se abordará la tarea de determinar la polaridad de las opiniones de los usuarios sobre lugares turísticos, centrándonos específicamente en restaurantes, hoteles y atracciones turísticas. Para ello utilizaremos el corpus Rest_Mex_2022 el está compuesto por un conjunto de datos de opiniones, este conjunto de datos contiene opiniones de usuarios sobre destinos turísticos, hoteles y restaurantes mexicanos. Las opiniones están escritas en español y están etiquetadas con su polaridad. Cada opinión en el corpus "Rest_Mex_2022" se evalúa con un valor numérico del 1 al 5, donde:

- 1 indica una opinión muy negativa.
- 2 indica una opinión negativa.
- 3 indica una opinión neutral.
- 4 indica una opinión positiva.
- 5 indica una opinión muy positiva.

El propósito principal de esta práctica es emplear el modelo de aprendizaje automático entrenado para predecir las opiniones contenidas en el conjunto de pruebas. El conjunto de pruebas proporciona una evaluación crítica del rendimiento del modelo, ya que incluye datos no vistos durante el proceso de entrenamiento. Al aplicar el modelo a este conjunto, buscamos verificar la capacidad del modelo para generalizar y realizar predicciones precisas sobre nuevas opiniones de usuarios en lugares turísticos.

Para poder realizar este proceso lo llevaremos a cabo mediante las siguientes etapas:

1. Procesamiento del Corpus
2. Representación del Texto
3. Entrenamiento de los Algoritmos de Machine Learning
4. Predicción

Metodología

Preprocesamiento del Corpus

Para realizar el preprocesamiento del corpus lo primero que hacemos después de cargar las bibliotecas y de haber cargado el corpus es una manipulación de los datos del dataframe esto para asegurarnos que los datos contenidos en las columnas son de tipo string esto con la finalidad de concatenar las columnas de title y Opinión en una sola a la cual le asignamos el nombre de texto. Pasando del corpus original (figura 1) al corpus con estas modificaciones (figura 2).

	Title	Opinion	Polarity	Attraction
0	Pésimo lugar	Piensen dos veces antes de ir a este hotel, te...	1	Hotel
1	No vayas a lugar de Eddie	Cuatro de nosotros fuimos recientemente a Eddi...	1	Restaurant
2	Mala relación calidad-precio	seguiré corta y simple: limpieza\n- bad. Tengo...	1	Hotel
3	Minusválido? ¡No te alojes aquí!	Al reservar un hotel con multipropiedad Mayan ...	1	Hotel
4	Es una porqueria no pierdan su tiempo	No pierdan su tiempo ni dinero, venimos porque...	1	Hotel

Figura 1. Corpus Original

	Polarity	Attraction	texto
0	1	Hotel	Pésimo lugar Piensen dos veces antes de ir a e...
1	1	Restaurant	No vayas a lugar de Eddie Cuatro de nosotros f...
2	1	Hotel	Mala relación calidad-precio seguiré corta y s...
3	1	Hotel	Minusválido? ¡No te alojes aquí! Al reservar u...
4	1	Hotel	Es una porqueria no pierdan su tiempo No pierd...

Figura 2. Corpus Modificado

Esta modificación al corpus se realizó para poder manipular el corpus mejor a futuro.

Posteriormente creamos una función denominada procesar_texto ya que a ese nuevo corpus lo sometemos a un procesamiento el cual contiene procesos de tokenizacion, lematización y eliminación de stopwords como se ha realizado en prácticas anteriores solo que ahora debido a que el texto del corpus se encuentra en español deberemos de cargar el idioma. (Figura 3)

```
# Cargar modelo de lenguaje español
nlp = spacy.load('es_core_news_sm')

def procesar_texto(texto):
    try:
        # Tokenización
        doc = nlp(texto)
        tokens = [token.text for token in doc]
        texto_tokenizado = " ".join(tokens)

        # Lematización
        doc = nlp(texto_tokenizado)
        lemas = [token.lemma_ for token in doc]
        texto_lematizado = " ".join(lemas)

        # Eliminación de stopwords
        doc = nlp(texto_lematizado)
        tokens_filtrados = [token.text for token in doc if token.pos_ not in ["DET", "ADP", "CCONJ", "PRON"]]
        texto_filtrado = " ".join(tokens_filtrados)
    except:
        # Ignorar contenido si no se puede procesar
        texto_filtrado = ""

    return texto_filtrado
```

Figura 3. Función de Procesamiento del Corpus

Iremos realizando este procesamiento iterando a través de la columna de texto de nuestro corpus y para ello iremos aplicando la función procesar_texto a cada elemento y almacenando los resultados en una lista llamada lista_contenido. (Figura 4)

```
lista_contenido = []

for i, contenido in enumerate(corpus['texto']):
    lista_contenido.append(procesar_texto(contenido))   # Normalizar contenido
    print(f"\rProgreso: {i/len(corpus):.1%}", end="")   # Mostrar porcentaje de avance
```

Figura 4. Procesamiento de la columna texto del corpus

Finalmente decidimos guardar el resultado de este procesamiento del corpus en archivo de tipo pickle con la finalidad de no estar repitiendo este proceso en cada ejecución.

Posteriormente se realizó la parte del entrenamiento y prueba utilizando el dataset compartido.

Primero se cargó el archivo pickle del corpus procesado anteriormente, además del corpus en Excel que contenía diferentes reseñas, aunque la columna que importaba era la de polaridad.

Para entrenar y probar los distintos algoritmos se solicitó que el conjunto de datos estuviera dividido en 80% datos de entrenamiento y el 20% restante en datos de prueba. Mediante la función “train_test_split” se obtienen los datos y las etiquetas del conjunto de entrenamiento y el conjunto de pruebas.

Después, mediante diferentes representaciones del texto (binario, frecuencia y tf-idf) se convirtió el texto de los datos del conjunto de entrenamiento y pruebas en vectores para así utilizar dichos vectores como entrada para entrenar y evaluar modelos de aprendizaje automático.

Una vez realizado esto se hizo verificamos si es que contamos con un diccionario de polaridades en caso de haberlo lo cargamos por otra parte si no hay ningún diccionario de polaridades llamamos a la función “load_sel()” en esta función crearemos un diccionario vacío de nombre “lexicon_sel” el cual será utilizado para almacenar las polaridades asociadas a cada palabra.

Luego abrimos el archivo llamado 'SEL_full.txt' en modo lectura ('r') el cual fue dado por el profesor y contiene información sobre polaridades de palabras. Posteriormente dividimos cada palabra usando el carácter de tabulación como separador y eliminamos el carácter de salto de línea ('\n') de la última palabra en la línea. Luego creamos una tupla llamada pair que contiene la palabra y su valor de polaridad. Esta tupla representa la información asociada a cada palabra en términos de polaridad.
En caso de que el diccionario este vacío creamos una lista que contiene la tupla pair y se asocia con la palabra clave en el diccionario.
Si ya contiene palabras, se agrega la tupla pair a la lista existente asociada con la palabra clave. Una vez que ya recorrimos todo el archivo eliminamos la entrada con la clave 'Palabra' del diccionario. Y regresamos el diccionario el cual se asigna el resultado a la variable lexicon_sel. Después de cargar o generar el diccionario, abre el archivo 'lexicon_sel.pkl' en modo de escritura binaria ('wb') y utiliza pickle.dump() para almacenar el contenido de lexicon_sel en el archivo.

Después, tomamos el conjunto de datos de entrenamiento y de prueba, así como el diccionario de polaridades lexicon_sel como entrada para la función getSELFeatures la cual sirve para acumula los valores de polaridad para seis emociones específicas: Alegría, Tristeza, Enojo, Repulsión, Miedo y Sorpresa. También crea un diccionario (dic) con claves que representan cada emoción y sus respectivos valores acumulados. Agrega dos claves adicionales al diccionario para los valores acumulados del mapeo a positivo (alegría + sorpresa) y negativo (enojo + miedo + repulsión + tristeza). Agrega el diccionario resultante a una lista llamada features para cada cadena en la entrada. El resultado, que es una lista de

diccionarios con características emocionales, se asigna a la variable polaridades_train y polaridades_test respectivamente.

Luego calcularemos las polaridades para los Conjuntos de Entrenamiento y Prueba utilizando la función de polaridad_cadenas. Llamamos a la función polaridad_cadenas con el conjunto de características emocionales de entrenamiento (polaridades_train) y prueba (polaridades_test) respectivamente.

La función polaridad_cadenas toma como entrada una lista de diccionarios (polaridades), donde cada diccionario representa las características emocionales de una cadena. Itera sobre cada elemento de la lista y extrae los valores acumulados de polaridad positiva (acumuladopositivo) y polaridad negativa (acumuladonegative) de cada diccionario. Construye un arreglo NumPy para cada cadena que contiene estos dos valores. Agrega estos arreglos al conjunto polaridad_cadenas. Devuelve la lista resultante la cual será guardada en las variables polaridad_cadenas_train y polaridad_cadenas_test respectivamente.

Después, convertimos las listas de arreglos de polaridades calculadas previamente en matrices esparzas comprimidas en formato CSR. Este formato es eficiente para almacenar matrices con una gran cantidad de ceros y las almacenamos en matriz_polaridades_train y matriz_polaridades_test. Finalmente, ampliamos las matrices originales de características (X_train_vector y X_test_vector) con las matrices de polaridades calculadas previamente (matriz_polaridades_train y matriz_polaridades_test).

Previamente a poner a entrenar los algoritmos de machine learning debemos de dividir el conjunto de entrenamiento en 5 pliegues por ello debemos de crear un objeto de validación cruzada estratificada utilizando la clase StratifiedKFold de la biblioteca scikit-learn.(Figura 5)

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=0)
```

Figura 5. Creación de objeto StratifiedKFold con 5 pliegues

Una vez realizado esto definimos los algoritmos de Machine Learning a utilizar cada uno con las características que determinamos eran las mejores los cuales fueron:

- Clasificador de Red Neuronal Perceptrón Multicapa (MLP).

```
MLPClassifier(hidden_layer_sizes=(50, 25), activation='relu', solver='adam',
  alpha=0.0001, learning_rate='constant', max_iter=200, batch_size='auto',
                          early_stopping=False)
```

- Clasificador de k-Vecinos más Cercanos.

```
KNeighborsClassifier(n_neighbors=3)
```

- Clasificador de Bosques Aleatorios.

```
RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
```

- Clasificador de Regresión Logística.

```
LogisticRegression(solver='lbfgs', max_iter=2000)
```

- Clasificador Naive Bayes multinomial.

```
MultinomialNB(alpha=0.1, fit_prior=True)
```

- GradientBoostingClassifier

```
GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3,
          min_samples_split=2, min_samples_leaf=1, subsample=1.0)
```

Debido a que ejecutar todos los algoritmos al mismo tiempo supone una complejidad y tiempo computacional alto fuimos probando cada algoritmo de Machine Learning por separado.

Una vez definido el algoritmo a realizar ejecutamos un proceso de entrenamiento y evaluación de un clasificador utilizando validación StratifiedKFold el cual previamente declaramos.

Lo primero que hacemos es iterar sobre cada fold generada por el StratifiedKFold. Los índices de entrenamiento y prueba para cada fold se obtienen en cada iteración. Luego dividimos el conjunto de datos de entrenamiento (X_train_vector e y_train) en conjuntos de entrenamiento (rasgos_entrenamiento y clases_entrenamiento) y prueba (rasgos_prueba y clases_prueba) utilizando los índices proporcionados por la validación cruzada. Después, entrenamos el clasificador utilizando los conjuntos de entrenamiento.
Después realizamos las predicciones sobre el conjunto de prueba (rasgos_prueba) utilizando el modelo entrenado.

Finalmente calculamos la puntuación F1 macro promedio sobre las clases de prueba y las clases predichas las cuales serán almacenadas en la lista f1_macro. Y para no perder esta métrica en las distintas ejecuciones de los algoritmos las guardamos en un archivo de texto para poder compáralas y decidir que algoritmo fue el que nos entregó mejores resultados que en nuestro caso fue el algoritmo de regresión logística como se puede ver en la Figura 6.

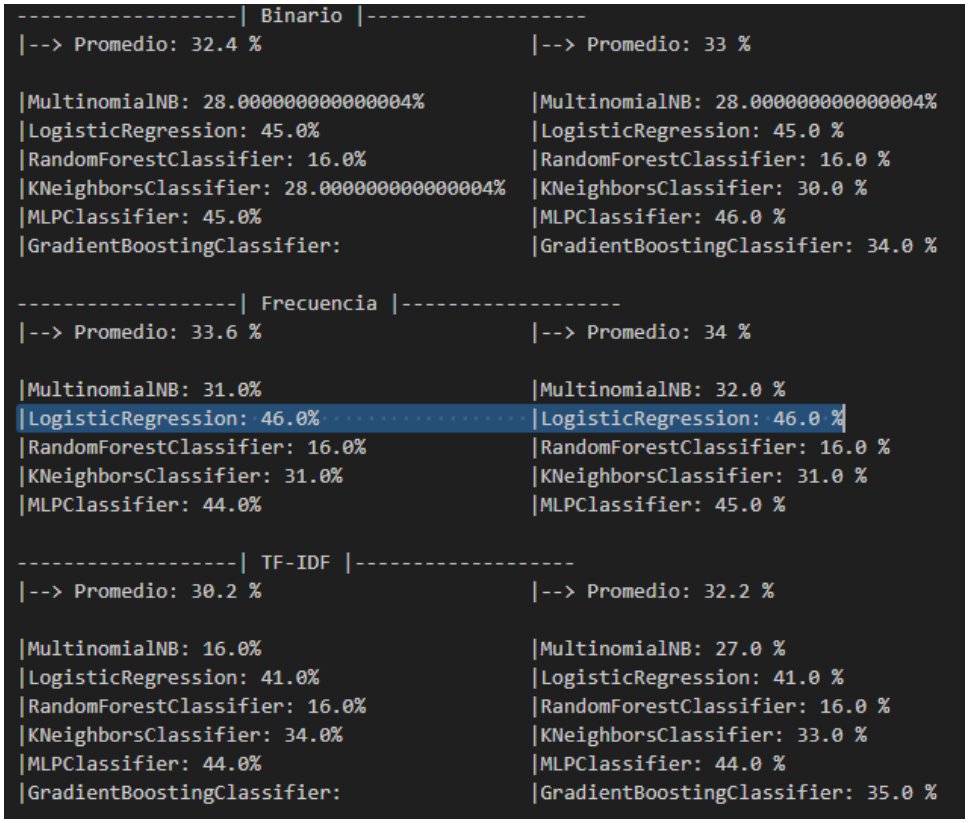
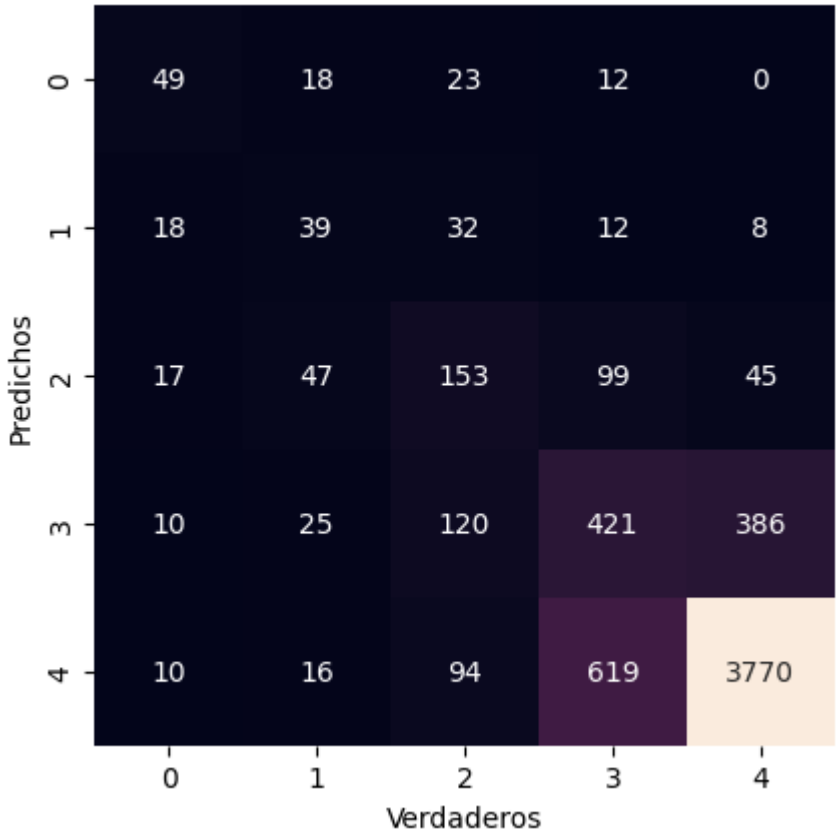


Figura 6. F1-Macro de los algoritmos de Machine Learning Ejecutados

Así que seleccionamos el algoritmo de Regresion Logistica para ser entrenado con el train set completo de igual manera se hizo el predict con el X test completo dándonos las siguientes métricas de evaluación:

	precision	recall	f1-score	support
1	0.48	0.47	0.48	104
2	0.36	0.27	0.31	145
3	0.42	0.36	0.39	422
4	0.44	0.36	0.40	1163
5	0.84	0.90	0.86	4209
accuracy			0.73	6043
macro avg	0.51	0.47	0.49	6043
weighted avg	0.71	0.73	0.72	6043

Siendo esta su matriz de confusión:



Propuestas de mejoras y trabajos a futuro

En cuanto a mejoras que se podrían realizar para mejorar el rendimiento del algoritmo que obtuvo mejor resultado podrían tomarse en cuenta las mismas recomendaciones que nos da la práctica, que son “Manejo de negación”, “Repetición de palabras” y “Corrector ortográfico”.

En la parte del manejo de negación en análisis de sentimientos es de gran ayuda para entender de forma concreta el significado de una recomendación si es que esta incluye demasiadas palabras de negación. Entre las estrategias se incluyen la manipulación de las palabras que siguen a estas recomendaciones negativas. Esto asegura que el modelo de análisis de sentimientos capture correctamente el cambio de polaridad asociado con las negaciones.

Para la repetición de palabras en el análisis de sentimientos quizás no sea de gran impacto, sin embargo, puede hacer una diferencia, ya que ciertas palabras pueden influir de gran manera en la predicción del sentimiento. Esto se podría lograr eliminando dicho problema en el preprocesamiento, con esto se garantizaría que a cada palabra se le asigne un peso equitativo en la evaluación del sentimiento sin tomar en cuenta la repetición de dicha palabra.

Por último, la corrección ortográfica ayuda a garantizar que una palabra sea interpretada o tomada en cuenta con el peso correcto, si se corrigen dichos errores se podría obtener un mejor sentido a la palabra y así obtener una mejor precisión en el análisis de sentimientos.

Conclusiones

Para concluir con el reporte, se mencionará lo más relevante acerca de lo hecho en la práctica. En primera parte, en cuanto al preprocesamiento del corpus no hay bastantes cambios a los esperados, ya que utilizamos el mismo proceso de práctica anteriores (tokenización, lematización y eliminación de stopwords), sin embargo, estos pasos son importantes para obtener los resultados obtenidos.

En cuanto al entrenamiento y prueba, la división del conjunto de datos en conjuntos de entrenamiento y prueba es demasiado común realizarlo de manera aleatoria, ya que al no hacerlo el resultado será muy poco favorable.

La parte que sorprendió en cuanto a los resultados obtenidos fue en la representación del texto mediante vectores (binaria, frecuencia y tf-idf), ya que en prácticas anteriores se había obtenido resultados poco favorables mediante la representación del vector por frecuencia, sin embargo, en esta práctica, los mejores resultados fueron por medio de dicha representación como se muestra en la Figura 6.

En cuanto a los algoritmos de Machine Learning, como se explicó anteriormente, se ocuparon MultinomialNB, LogisticRegression, RandomForestClassifier, KNeighborsClassiffier, MLPClassifier y GradientBoostingClassifier. Como se mostró, el algoritmo que brindó mejores resultados fue el de Regresión Logística, pero este tardaba un tiempo si bien no tan amplio tampoco fue muy veloz. Pero, si quisiéramos mayor velocidad en vez de mayor precisión se podría escoger el algoritmo de MLPClassifier que obtenía resultados bastantes buenos en poco tiempo.

Con lo mencionado anteriormente, concluimos que el proceso aborda de gran manera la tarea de análisis de sentimientos, desde el preprocesamiento de datos hasta la evaluación de modelos, y con estos se puede obtener la mejor elección de algoritmos para obtener buenos resultados.