Tarea 3 – ISIS 4221

Natural Language Processing

**Integrantes:**

Santiago Martínez Cód. 201533279

Camilo Castañeda Cód. 202314092

## Clasificadores (NB y LR):

Respecto a la comparación de *Naive Bayes* (NB) y *Regresión Logística* (LR), en el conjunto de datos 20N para identificar las 20 categorías diferentes de *newsgroups*, lográndolo mediante el entrenamiento de estos dos clasificadores en diferentes representaciones de texto (**tf y tfidf**) y la evaluación de su rendimiento en conjuntos de entrenamiento, validación y prueba.

## Pipeline de procesamiento:

Se realiza un proceso completo de procesamiento de texto que incluye la creación de representaciones de texto (**tf y tfidf**) y la capacitación de los clasificadores (NB y LR) en cada una de estas representaciones, y se asemeja a la creación de un pipeline de procesamiento de datos para la tarea de clasificación de texto.

En el siguiente código se puede observar a la definición de los vectorizadores y la transformación de los conjuntos de datos de entrenamiento, validación y prueba:

# Define vectorizers for "tf" and "tfidf" representations

tf\_vectorizer = CountVectorizer(max\_features=5000, stop\_words="english")

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000, stop\_words="english")

# Fit and transform the vectorizers on training data

X\_train\_tf = tf\_vectorizer.fit\_transform(X\_train)

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

# Transform validation and test data

X\_val\_tf = tf\_vectorizer.transform(X\_val)

X\_val\_tfidf = tfidf\_vectorizer.transform(X\_val)

X\_test\_tf = tf\_vectorizer.transform(X\_test)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

## Entrenamiento de NB y LR:

Se entrena tanto el clasificador *Naive Bayes* como el *Regresión Logística* utilizando dos representaciones de texto diferentes: "**tf**" (frecuencia de términos) y "**tfidf**" (frecuencia de términos ponderada por la frecuencia inversa de documentos). Luego, evalúa el rendimiento de estos clasificadores en los conjuntos de validación y prueba.

Para la creación de las representaciones se implementan los siguientes métodos de la librería **sklearn** *CountVectorizer* que crea una matriz de conteo de palabras a partir de documentos de texto. Se utiliza para la representación "**tf**". Y *TfidfVectorizer* que crea una matriz **TF-IDF** a partir de documentos de texto. Además, se utiliza para la representación "**tfidf**".

## Técnica de validación cruzada:

Se utiliza la técnica de validación cruzada de 10 pliegues (KFold con n\_splits=10) para evaluar el rendimiento de los clasificadores NB y LR. Respecto a la validación cruzada la definimos como una estrategia usada para evaluar qué tan bien generaliza un modelo de aprendizaje automático. Y consiste en dividir el conjunto de datos en múltiples subconjuntos (pliegues), entrenar el modelo en algunos de estos subconjuntos y probarlo en los subconjuntos restantes, se repite varias veces y las métricas de rendimiento se promedian en estas iteraciones.

En la siguiente sección del código comienza la definición de los folds y la creación del objeto KFold:

# Define the number of folds for cross-validation

n\_folds = 10

kf = KFold(n\_splits=n\_folds, shuffle=True, random\_state=13)

## Comparación de resultados de NB y LR utilizando 10-fold cross-validation:

Se utilizan los conjuntos de entrenamiento y validación combinados para realizar la validación cruzada en lugar de usar solo el conjunto de entrenamiento. También se realiza una búsqueda de hiperparámetros para Regresión Logística (**LR**) utilizando *GridSearchCV*. Esto se hace para encontrar los mejores hiperparámetros para el modelo **LR**, incluido el parámetro de regularización C y el algoritmo del solucionador. Finalmente, los resultados de validación cruzada se almacenan en las variables "*nb\_cv\_preds*" y "*lr\_preds*" para Naive Bayes y Regresión Logística, respectivamente.

## Reporte de métricas:

Se genera informes de clasificación que incluyen *precisión, recall y puntuaciones F1* para cada clase y estadísticas agregadas, como macro y micro promedios. Los informes de clasificación se generan tanto para Naive Bayes como para Regresión Logística después de la validación cruzada.

## Reporte de métricas en el conjunto de prueba:

Se evalúan los modelos NB y LR en el conjunto de prueba utilizando las representaciones "**tfidf**". Además, se genera informes de clasificación que incluyen estas métricas tanto para NB y RL.

## Identificación del mejor modelo:

Para la determinación de cuál es el mejor modelo entre NB y LR, es necesario comparar sus resultados en el conjunto de prueba, se considera que el mejor modelo es aquel que obtiene las métricas más altas en el conjunto de prueba, en términos de precisión, recall y puntuaciones F1. Los resultados de ambas modelos en el conjunto de prueba se pueden comparar para tomar una decisión sobre cuál de los dos modelos es el mejor en función de estas métricas.

En la siguiente sección del código se informan las métricas de *precisión, recall y F1* para el mejor modelo de Regresión Logística (LR), en seguida la validación cruzada, luego se realiza una búsqueda de hiperparámetros para el modelo de NB y se realiza la validación cruzada de 10 folds con el mejor modelo. Finalmente, se informan las métricas de *precisión, recall y F1* para el modelo de NB en el conjunto combinado de entrenamiento y validación.

# Report precision, recall, and F1 scores for the best Logistic Regression model

lr\_preds = best\_lr\_classifier.predict(X\_combined)

print("Logistic Regression Classification Report (Combined Training+Validation Set):")

print(classification\_report(y\_combined, lr\_preds))

# Hyperparameter search for Naive Bayes (alpha parameter)

nb\_param\_grid = {'alpha': [0.1, 0.5, 1.0, 2.0]}  # parámetro para regular suavizado laplaciano

nb\_grid\_search = GridSearchCV(nb\_classifier, nb\_param\_grid, cv=kf, scoring='f1\_macro')

nb\_grid\_search.fit(X\_combined, y\_combined)

best\_nb\_classifier = nb\_grid\_search.best\_estimator\_

# Perform 10-fold cross-validation with the best Naive Bayes classifier

nb\_cv\_preds = cross\_val\_predict(best\_nb\_classifier, X\_combined, y\_combined, cv=kf)

# Report precision, recall, and F1 scores for Naive Bayes

print("Naive Bayes Classification Report (Training Set + Validation Set):")

print(classification\_report(y\_combined, nb\_cv\_preds))

## Resultados de Validación:

**NB**: Precisión del 93.23%

**LR**: Precisión del 94.56%

## Resultados de Prueba:

**NB**: Precisión del 87.40%

**LR**: Precisión del 87.67%

En ambos conjuntos, tanto validación y como prueba, la **LR** supera a **NB** en términos de precisión. Esto indica que, en promedio, **LR** clasifica correctamente un mayor porcentaje de documentos en sus respectivas categorías en comparación con **NB** cuando se utiliza la representación **TF-IDF**.

# **Análisis de sentimientos**

Se utiliza el conjunto de datos "*Multi-Domain Sentiment Dataset*" para construir un clasificador de sentimientos (positivos/negativos) para cada una de las categorías mencionadas ("Books", "DVD", "Electronics", "Kitchen").

Se combina los *negative review & positive review* de cada categoría para formar un conjunto de entrenamiento y validación para esa categoría específica. Esto se realiza en un bucle que recorre todas las categorías. Después se procede a utilizar estas revisiones combinadas como el conjunto de datos de entrenamiento y validación para entrenar y evaluar clasificadores de sentimientos basados en los algoritmos NB y LR.

A continuación, se muestra la sección inicial del código:

for category in categories:

    positive\_reviews, positive\_labels = extract\_reviews\_and\_labels(category, "positive", max\_reviews=None)

    negative\_reviews, negative\_labels = extract\_reviews\_and\_labels(category, "negative", max\_reviews=None)

    X\_test, y\_test = extract\_reviews\_and\_labels(category, "unlabeled", max\_reviews=None)

    X\_train = positive\_reviews + negative\_reviews

    y\_train = positive\_labels + negative\_labels

Se carga el conjunto de datos de prueba (revisiones no etiquetadas) de cada categoría y se utiliza como datos de prueba en el bloque de código que realiza el análisis de sentimiento en todo el conjunto de datos combinado.

Se realiza el análisis de sentimiento utilizando NB y LR como algoritmos de clasificación en el conjunto de prueba. Se calcula las métricas de *precisión, recall, F1 y exactitud* para cada combinación de algoritmo y representación de características (**tf y tfidf**). Los resultados se almacenan en el *DataFrame results\_df*, que luego se muestra en la salida.

En la siguiente sección de código se muestra lo comentado en el párrafo anterior:

# Perform sentiment analysis on the merged dataset for NB and LR

for algorithm in ["NB", "LR"]:

    for representation, vectorizer in vectorizers.items():

        print(f"Algorithm: {algorithm}, Representation: {representation}")

        # Initialize the vectorizer

        X\_train\_vectorized = vectorizer.fit\_transform(X\_train)

        X\_test\_vectorized = vectorizer.transform(X\_test)

        # Train the classifier

        if algorithm == "NB":

            classifier = MultinomialNB() ## TODO algun procesamiento para usar lexicones

        elif algorithm == "LR":

            classifier = LogisticRegression(max\_iter=500, random\_state=13)

        classifier.fit(X\_train\_vectorized, y\_train)

        # Predict sentiment

        y\_pred = classifier.predict(X\_test\_vectorized)

        # Map class labels to numerical values

        label\_mapping = {'negative': 0, 'positive': 1}

        y\_train\_numeric = [label\_mapping[label] for label in y\_train]

        y\_test\_numeric = [label\_mapping[label] for label in y\_test]

        # Calculate evaluation metrics

        precision = precision\_score(y\_test, y\_pred, average="weighted")

        recall = recall\_score(y\_test, y\_pred, average="weighted")

        f1 = f1\_score(y\_test, y\_pred, average="weighted")

        accuracy = accuracy\_score(y\_test, y\_pred)

        # Store results in the dataframe

        results\_df = results\_df.append(

            {

                "Algorithm": algorithm,

                "Representation": representation,

                "Precision": precision,

                "Recall": recall,

                "F1": f1,

                "Accuracy": accuracy,

            },

            ignore\_index=True,

        )

Features only extracted from lexicons1. Please document which features you built with enough detail (Pendiente)

## NB respecto a LR:

Tanto NB como LR muestran un desempeño bastante similar en términos de las métricas de *precisión, recall, F1 y exactitud*. En el conjunto de datos combinado dónde se juntan todas las categorías, LR tiende a superar ligeramente a NB en términos de F1 y precisión.

## Categorías ("Books", "DVD", "Electronics", "Kitchen"):

Según la métrica F1, las categorías más difíciles de clasificar parecen ser "DVD" y "Books". En estas categorías independientemente de la representación o el algoritmo de clasificación, el rendimiento es menor en comparación con las categorías "Electronics" y "Kitchen".

       Category Algorithm Representation  Precision    Recall        F1  \

0         books        NB             tf   0.834947  0.826652  0.825808

1         books        NB          tfidf   0.839174  0.823964  0.822292

2         books        LR             tf   0.824635  0.824636  0.824635

3         books        LR          tfidf   0.833220  0.832027  0.831960

4           dvd        NB             tf   0.820063  0.819855  0.819847

5           dvd        NB          tfidf   0.848160  0.846068  0.845892

6           dvd        LR             tf   0.832617  0.832404  0.832355

7           dvd        LR          tfidf   0.845131  0.844953  0.844914

8   electronics        NB             tf   0.854781  0.854779  0.854777

9   electronics        NB          tfidf   0.866608  0.865517  0.865441

10  electronics        LR             tf   0.858618  0.858476  0.858451

11  electronics        LR          tfidf   0.859836  0.859708  0.859685

12      kitchen        NB             tf   0.877945  0.877881  0.877869

13      kitchen        NB          tfidf   0.880580  0.880404  0.880380

14      kitchen        LR             tf   0.881959  0.881918  0.881910

15      kitchen        LR          tfidf   0.868941  0.868797  0.868775

## Entrenamiento de un único clasificador:

Entrenar un único clasificador para todas las categorías en lugar de tener un clasificador separado para cada categoría porque simplifica el proceso y permite que el clasificador aprenda patrones comunes en todas las categorías.

## Evaluación y comparación de resultados:

Para NB:

* La representación **tfidf** supera ligeramente a la representación **tf** en todas las métricas.
* La precisión global está en el rango del 86.05% al 86.85%.

Para LR:

* La representación **tf** supera ligeramente a la representación **tfidf** en todas las métricas.
* La precisión global está en el rango del 86.76% al 87.62%.

Comparación entre algoritmos:

* En términos de *precisión y F1*, LR supera a NB para ambas representaciones de características.
* La representación **tf** supera ligeramente a **tfidf** para ambos algoritmos.

Características importantes:

* Para LR con representación **tf**, las 10 características más importantes incluyen palabras como "easy," "wonderful," "amazing," "love," "the\_best," "fantastic," "best," "perfect," "great," y "excellent."
* Para LR con representación **tfidf**, las 10 características más importantes incluyen palabras como "well," "price," "easy\_to," "perfect," "the\_best," "easy," "love," "best," "excellent," y "great."