Práctica 2

Grado en Ciencia e Ingeniería de Datos

Procesamiento de Lenguaje Natural 2025-26

Clasificación de texto para análisis de sentimiento

Pareja 01 → Héctor Tablero Díaz y Álvaro Martínez Gamo.

1. Trabajo Previo (Preprocesamiento)

Para la preparación de los datos textuales, se aplicó un pipeline de preprocesamiento específico con el objetivo de limpiar, normalizar y estandarizar el corpus antes de la extracción de características.

El proceso de preprocesamiento ejecutado consistió en los siguientes pasos secuenciales:

1. `remove_html`: Eliminación de cualquier etiqueta o artefacto HTML presente en el texto.
2. `remove_urls`: Detección y eliminación de direcciones web (URLs).
3. `lowercase`: Conversión de todo el texto a minúsculas para unificar el vocabulario.
4. `remove_extra_whitespace`: Eliminación de espacios en blanco redundantes (espacios múltiples, saltos de línea, tabulaciones) para normalizar el espaciado.
5. `remove_stopwords`: Filtrado y eliminación de palabras comunes (ej. "el", "la", "de") que no aportan valor semántico significativo para la clasificación.
6. `lemmatize`: Lematización del texto, reduciendo las palabras a su forma raíz o lema (ej. "corriendo" → "correr"). Este paso es fundamental para agrupar términos conceptualmente similares y reducir la dimensionalidad del vocabulario.

2. Representación Vectorial

Se exploraron y evaluaron múltiples esquemas de representación vectorial para transformar el texto preprocesado en un formato numérico comprensible por los algoritmos de aprendizaje automático.

Como se pueden ver en el `informe_tecnico.txt`, las representaciones utilizadas fueron:

- **TF-IDF (Term Frequency-Inverse Document Frequency):** Se generaron vectores basados en la frecuencia de términos ajustada por su rareza en el corpus. Se probaron tres variantes limitadas por el número de características:
  - `tfidf_1gram`: 5000 características (unigramas).
  - `tfidf_2gram`: 10000 características (bigramas).
  - `tfidf_3gram`: 15000 características (trigramas).
- **Características Lingüísticas (`linguistic`):** Se diseñó un conjunto de 23 características de ingeniería de variables (*feature engineering*). Estas características capturan propiedades metatextuales y estilísticas del texto (ej. longitud de la reseña, uso de negaciones, recuento de signos de puntuación) en lugar de su contenido semántico.
- **Combinadas:** Se crearon representaciones híbridas concatenando los vectores TF-IDF con el vector de 23 características lingüísticas (ej. `combined_1gram` con 5023 características).

3. Modelos

Para la tarea de clasificación de polaridad (negativa, neutral, positiva), se seleccionaron dos algoritmos de aprendizaje automático robustos y comúnmente utilizados, como se observa en el archivo `evaluation_results.csv`:

- **Random Forest (RF):** Un modelo de ensamblaje basado en árboles de decisión. Se eligió por su alta precisión, su capacidad para manejar diferentes tipos de características (numéricas y categóricas, aunque aquí todas son numéricas) y su robustez frente al sobreajuste. Es particularmente efectivo con datos tabulares, como el conjunto de características lingüísticas.

- **Support Vector Machine (SVM):** Un clasificador de margen máximo. Se seleccionó por su rendimiento demostrado en espacios de características de alta dimensión y dispersos, como los generados por TF-IDF.

4. Evaluación

La metodología de evaluación se diseñó para asegurar la fiabilidad y generalización de los resultados, utilizando una partición estricta de los datos.

- **División de Datos (Data Split):** El corpus total se dividió en tres conjuntos, según el `informe_tecnico.txt`:
  - **Entrenamiento (Train):** 270,765 documentos.
  - **Validación (Validation):** 67,689 documentos.
  - **Pruebas (Test):** 84,612 documentos.
- **Proceso:** Los modelos se entrenaron con el conjunto de *Train*. El conjunto de *Validation* se utilizó exclusivamente para la optimización de hiperparámetros. Finalmente, el rendimiento del mejor modelo se reportó sobre el conjunto de *Test*, que el modelo no había visto previamente.
- **Métricas:** El rendimiento se midió utilizando métricas estándar de clasificación:
  - **Accuracy:** Proporción de clasificaciones correctas.
  - **Precision:** Capacidad del modelo para no etiquetar como positiva una muestra que es negativa.
  - **Recall:** Capacidad del modelo para encontrar todas las muestras positivas.
  - **F1-Score:** Media armónica de Precision y Recall, utilizada como la métrica principal para la optimización (específicamente `f1_cv` y `f1_test`).

5. Optimización de Hiperparámetros

Se realizó un proceso sistemático de ajuste de hiperparámetros para encontrar la configuración óptima de cada combinación de modelo y representación. El archivo `evaluation_results.csv` documenta este proceso, mostrando los `best_params` encontrados.

Este proceso utilizó el conjunto de validación y la métrica F1-Score (cross-validation) para identificar los parámetros que maximizaban el rendimiento.

Por ejemplo, para el modelo SVM se optimizaron parámetros como `C`, `loss` y `max_iter`, mientras que para Random Forest se ajustaron `n_estimators`, `min_samples_split`, `min_samples_leaf` y `max_depth`.

6. Resultados y Análisis

La evaluación comparativa de todas las combinaciones de modelos y representaciones se llevó a cabo utilizando el F1-Score en el conjunto de *Test* como métrica principal. Los resultados consolidados se presentan en la siguiente tabla.

Resultados de Evaluación (Ranking)

Rank	Modelo	Representación	F1-Test
1	**Random Forest**	**Características Lingüísticas**	**1.0000**
2	Random Forest	Unigramas + Lingüísticas	0.9955
3	SVM	Características Lingüísticas	0.9933
4	SVM	Unigramas + Lingüísticas	0.9933
5	SVM	Bigramas + Lingüísticas	0.9926

Análisis del Mejor Modelo

Los resultados identifican de manera concluyente al **Random Forest** que utiliza únicamente las **Características Lingüísticas** (el vector de 23 características) como el modelo de rendimiento superior.

- **Hiperparámetros Óptimos:** Los parámetros que produjeron este resultado fueron `{'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_depth': 10}`.
- **Rendimiento en Test:** Este modelo alcanzó un rendimiento perfecto en el conjunto de pruebas, logrando un 100% en todas las métricas:
  - Accuracy: 1.0000
  - Precision (ponderada): 1.0000

    - Recall (ponderada): 1.0000
    - F1-Score (ponderado): 1.0000

    * El accuracy real es 0.999988181, pero al acotarlo al 5 decimales con redondeo, se obtiene el 1.0000.

- **Rendimiento por Clase:** El informe detallado y la matriz de confusión confirman que el modelo clasificó correctamente todas las instancias de las tres clases (negativa, neutral y positiva), con la excepción de un único error (una instancia negativa predicha como neutral).

Real / Predicho	neg	neu	pos
neg	28203	1	0
neu	0	28204	0
pos	0	0	28204

- **Generalización:** El modelo obtuvo métricas perfectas (1.0000) solo en el conjunto de *Test*, mientras que en los conjuntos de *Train* y *Validation* tuvo un 0.9259. Esto nos da a entender que generaliza bien, y que ha sido una coincidencia que el valor de *Test* haya superado al de *Validation*.

Discusión

El hallazgo más significativo es que la representación más simple (23 características lingüísticas) superó drásticamente a todas las representaciones semánticas (TF-IDF) y combinadas. Esto sugiere que, para este conjunto de datos específico, las características de ingeniería de variables (como la longitud del texto, puntuación, etc.) contienen una señal tan fuerte que son capaces de predecir la polaridad de la reseña con una exactitud casi perfecta.

Un F1-Score de 1.0000 es un resultado atípico en tareas de PLN. Indica que las 23 características lingüísticas son predictores perfectos para las clases. Si bien es un éxito desde la perspectiva de las métricas, en un escenario real, esto podría justificar una revisión para descartar una posible fuga de datos (*data leakage*), donde alguna de las 23 características esté inadvertidamente codificando la etiqueta de clase.

Nosotros hemos investigado a fondo, pero la lista de features claramente excluye `doc_id` y `label`, por lo que este no parece ser el problema. Además, si se hubiese filtrado alguno de estos datos, como mínimo todos los árboles de decisión deberían haber acertado el 100% de las veces, y esto no es el caso ni en *Train*/*Validation* ni en todos los conjuntos de características.