

PROYECTO 1

Pipeline de NLP | Grupo 7

Diego Leiva

Pablo Orellana

Renatto Guzman



Objetivos

Desarrollar un pipeline completo de Procesamiento de Lenguaje Natural (PLN) para clasificar automáticamente reseñas de películas en español según su polaridad (positiva o negativa).

- Preprocesar y limpiar un corpus en español.
- Representar el texto mediante técnicas clásicas y modernas.
- Entrenar y evaluar modelos probabilísticos y de clasificación.
- Comparar métricas de desempeño y visualizar resultados.

Preprocesamiento del Corpus

- ★ Corpus:
 - Dataset de reseñas en español del proyecto IMDB Dataset of 50K Movie Reviews (Kaggle).



Proceso:

- Eliminación de columnas innecesarias y duplicados.
- Conversión de etiquetas de sentimiento a formato binario (positivo=1, negativo=0).
- Limpieza y normalización:
 - Conversión a minúsculas.
 - Eliminación de tildes, URLs, correos, hashtags y puntuación.
 - Remoción de números y espacios múltiples.
- Tokenización y stemming con NLTK (SpanishStemmer).

Preprocesamiento del Corpus

Levenshtein

Sirve para detectar similitudes y errores ortográficos. No se aplica a todo el dataset debido a que:

- El corpus tiene ~50 000 reseñas → millones de palabras únicas.
- Comparar cada palabra con todas las demás sería $(\Theta(n^2))$, ineficiente y sin valor práctico.



Ejemplos de correcciones detectadas:

gargle → gaggle

cued → cue

tielos → cielos

louese → louise

bitmy → bitty

focales → vocales

montanosos → montanoso

coarse → corse

opondran → pondran

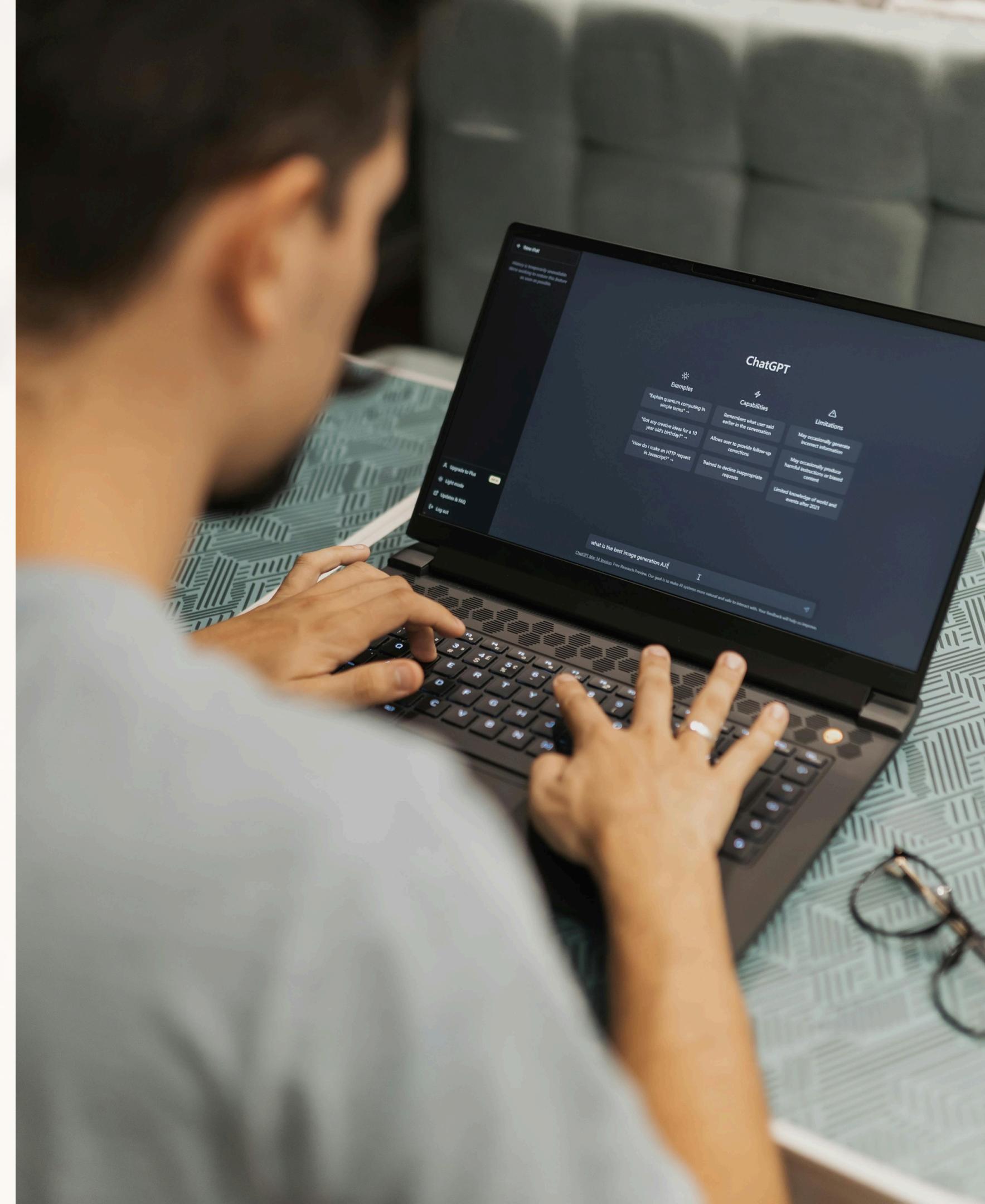
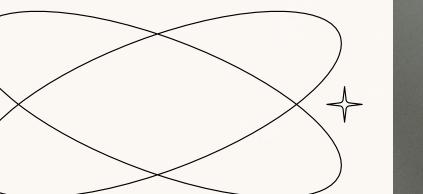
lagdon → lagoon



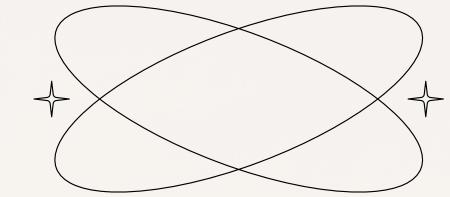
Representación del Texto

Técnicas utilizadas:

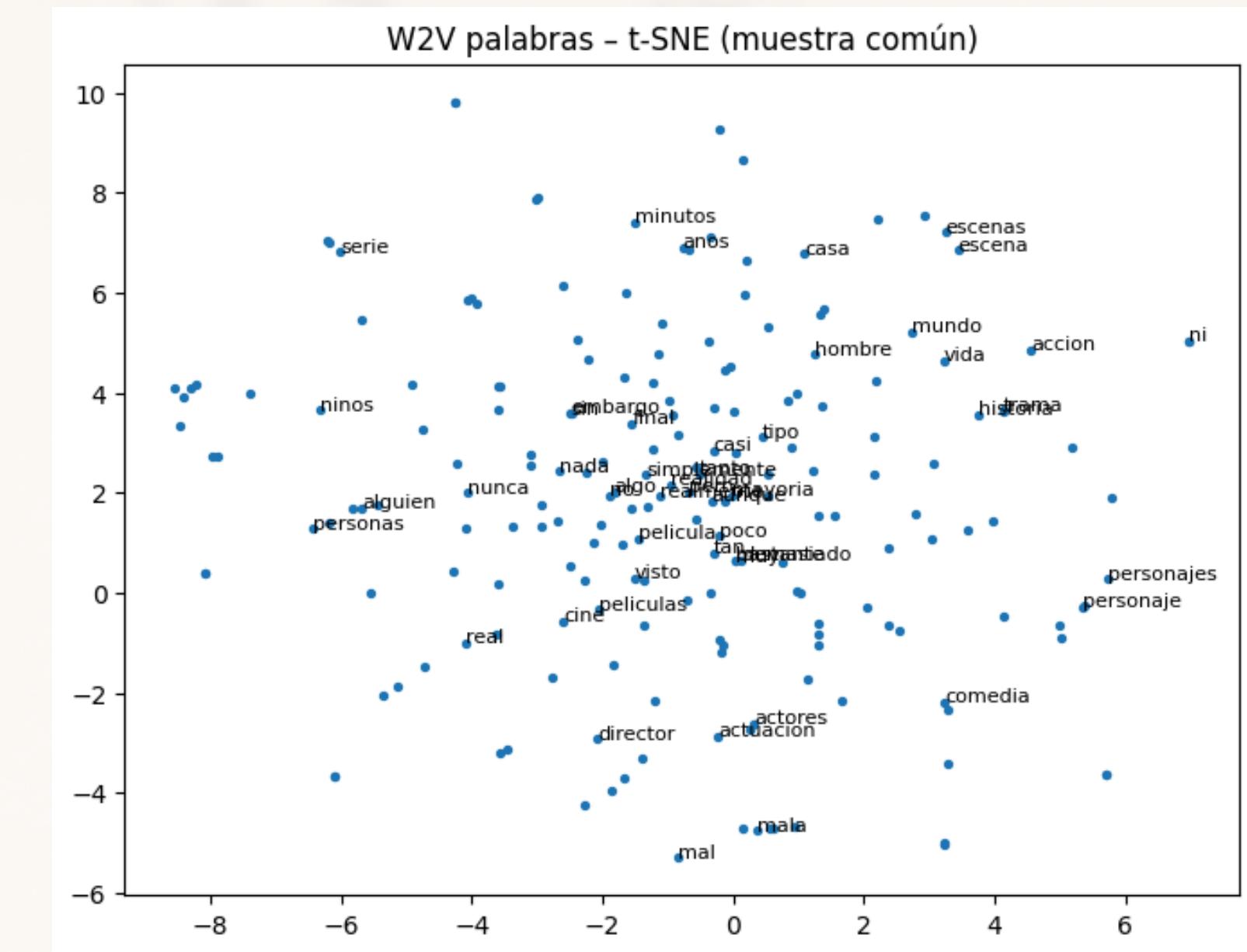
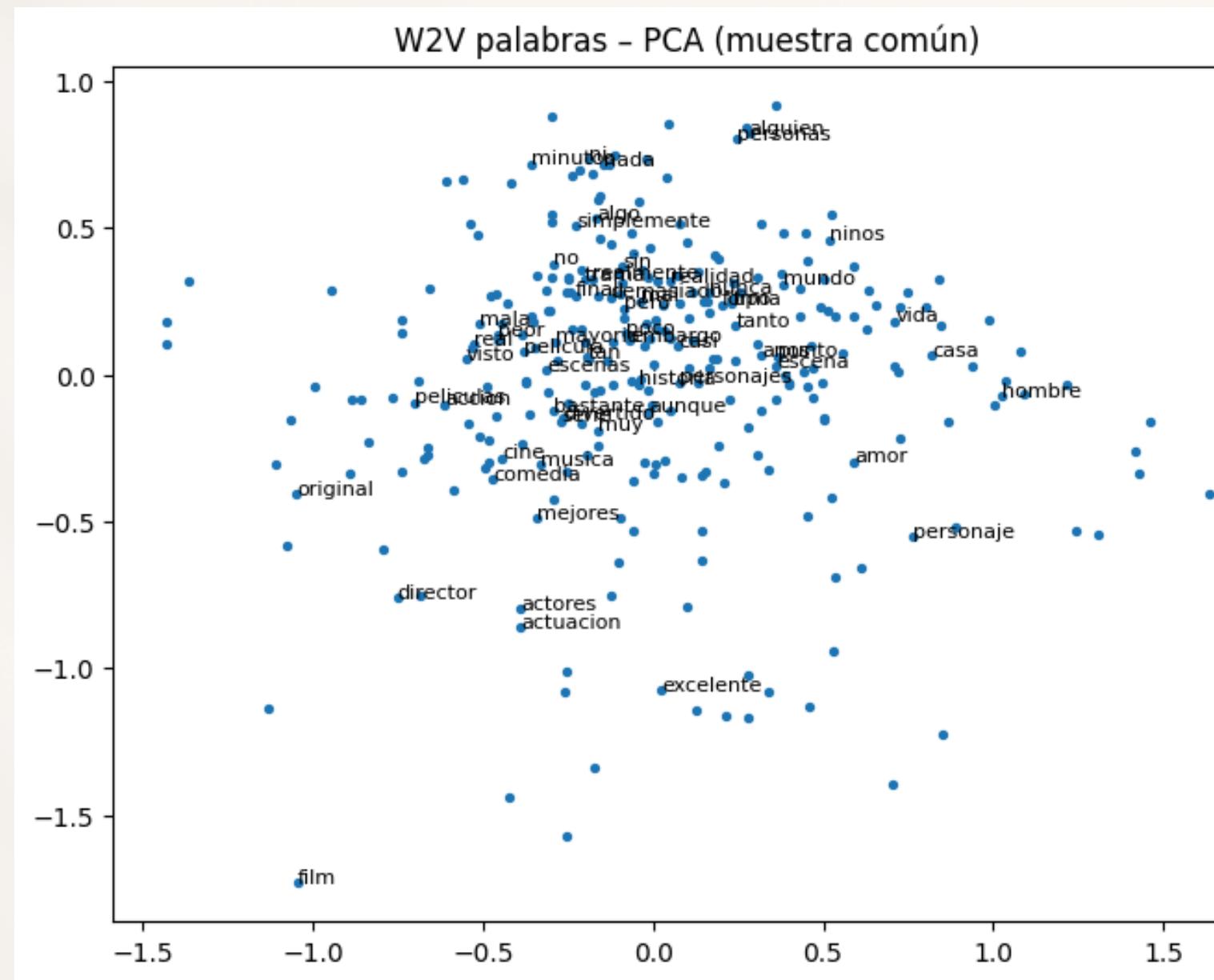
- Bag of Words (BoW) y TF-IDF con CountVectorizer y TfidfVectorizer.
- Generación de matriz de co-ocurrencia y aplicación de PPMI (Positive Pointwise Mutual Information).
- Entrenamiento de Word2Vec para generar embeddings semánticos.



Representación del Texto



Reducción de dimensionalidad con PCA y t-SNE para observar agrupaciones de palabras con significado similar.



Modelos Probabilísticos

Implementación:

- Modelo de lenguaje basado en bigramas y trigramas.
- Cálculo de entropía y perplejidad para evaluar la fluidez del modelo.
- Uso de técnicas de smoothing (Laplace) para manejar secuencias poco frecuentes.

```
== N-gramas (entrenamiento) ==
Vocab: 80242
Unigramas: 80,242
Bigramas: 2,507,851
Trigramas: 3,872,405
```

```
== Entropía / Perplejidad ==
Tokens evaluados: 1,044,505
Bigram Add-k (k=0.1) -> H=13.450 bits, PP=11191.71
Trigram Add-k (k=0.1) -> H=15.849 bits, PP=59040.98
Bigram Kneser-Ney -> H=12.642 bits, PP=6389.73
```

Ejemplos de probabilidad de oraciones:

```
--- Oración: la pelicula fue excelente
Bigram Add-k H=7.458, PP=175.80
Trigram Add-k H=11.322, PP=2559.64
Bigram KN      H=7.799, PP=222.76
```

```
--- Oración: la trama resulto aburrida
Bigram Add-k H=10.409, PP=1359.84
Trigram Add-k H=16.310, PP=81252.76
Bigram KN      H=10.115, PP=1109.24
```

Modelo Avanzado - Clasificación Supervisada

Modelos implementados:

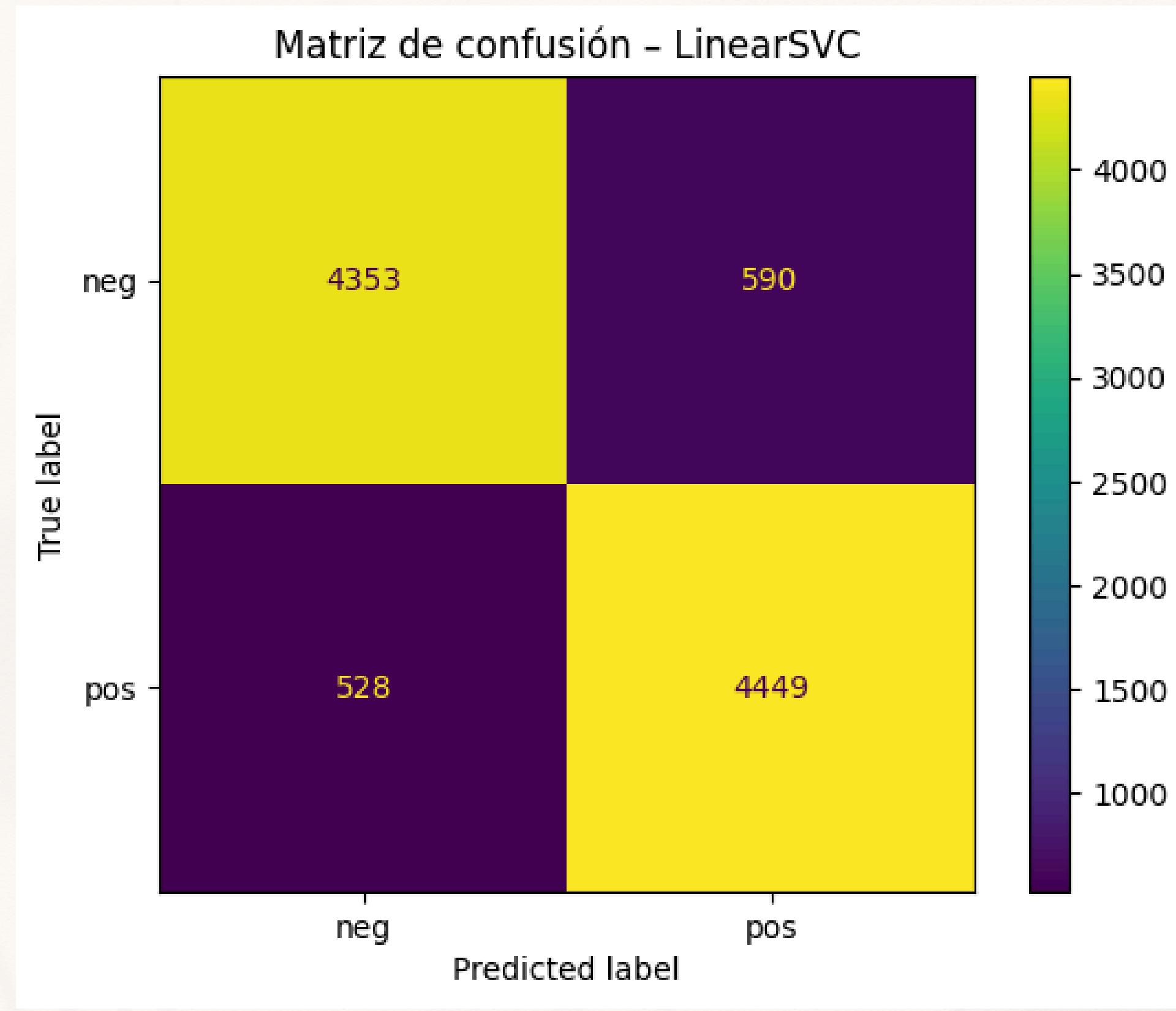
- Naive Bayes Multinomial
- Support Vector Machine (SVM lineal)

LinearSVC (TF-IDF palabras+caracteres)				
Accuracy: 0.8873 F1: 0.8884				
	precision	recall	f1-score	support
0	0.8918	0.8806	0.8862	4943
1	0.8829	0.8939	0.8884	4977
accuracy			0.8873	9920
macro avg	0.8874	0.8873	0.8873	9920
weighted avg	0.8874	0.8873	0.8873	9920

MultinomialNB (TF-IDF palabras+caracteres)				
Accuracy: 0.855 F1: 0.8558				
	precision	recall	f1-score	support
0	0.8558	0.8527	0.8543	4943
1	0.8543	0.8573	0.8558	4977
accuracy			0.8550	9920
macro avg	0.8550	0.8550	0.8550	9920
weighted avg	0.8550	0.8550	0.8550	9920

Mejor modelo baseado en F1

★
★
★



GRACIAS