UNIVERSIDAD DEL VALLE DE GUATEMALA

Procesamiento de Lenguaje Natural

Sección 20

Ing. Luis Esturbán

Proyecto 1: Pipeline completo de NLP

Integrantes:

- Pablo Orellana
- Diego Leiva
- Renatto Guzmán

Guatemala, 05 de octubre de 2025

1. Introducción

Este trabajo construye y evalúa un sistema de clasificación supervisada de sentimiento para reseñas de películas en español. Se compara un clasificador lineal SVM y un Naive Bayes Multinomial sobre representaciones TF-IDF de palabras y de caracteres. Además, se explora modelado probabilístico con n-gramas y una representación semántica con Word2Vec. Se reportan métricas estándar, matrices de confusión y análisis crítico.

2. Justificación del dataset

Se utiliza **IMDB Dataset of 50K Movie Reviews – Spanish**, una versión en español de 50 000 reseñas con etiquetas binarias {positivo, negativo}.

Ventajas:

- Tamaño suficiente para entrenar y validar modelos clásicos.
- Dominio estable y ampliamente estudiado, útil para comparar contra referencias.
- Etiquetado balanceado a gran escala (supuesto común en IMDB), lo que reduce sesgos de clase.

Riesgos y supuestos:

- Posible ruido de traducción o variación dialectal del español.
- Reseñas centradas en cine; la generalización a otros dominios es limitada.
- Etiquetado automático en variantes del dataset puede introducir ruido.

3. Metodología

3.1. Preparación del corpus

- Carga del CSV y depuración: eliminación de columnas auxiliares, rename(review_es→review), mapeo de etiquetas positivo→1, negativo→0, drop_duplicates, dropna.
- División posterior estratificada 80/20 para evaluación.

3.2. Normalización y limpieza

- Minúsculas y *strip*.
- Eliminación de diacríticos con *unicode NFD*.
- Remoción de URLs, correos, menciones y hashtags.
- Eliminación de puntuación, guiones bajos y dígitos; colapsado de espacios.

3.3. Stopwords con protección semántica

- Base NLTK (es/en) + listas externas.
- **Protección**: negaciones, conectores contrastivos e intensificadores no se eliminan.
- Aplicación con expresión regular basada en bordes de palabra.
- Filtrado de textos vacíos tras limpieza.

3.4. Tokenización y *stemming*

- nltk.word_tokenize (idioma: es)
- *SpanishSnowballStemmer*.
- Persistencia de: tokens y tokens_stem.

3.5. Corrección ortográfica puntual (opcional)

- Implementación de distancia de **Levenshtein**.
- Se usa para inspección y ejemplos sobre vocabulario raro; **no** se aplica masivamente por costo (O(n^2)).

3.6. Representaciones

- **Conteo y TF-IDF**
  - Texto con *stems* para palabras; texto sin stopwords para caracteres.
  - **BoW / TF-IDF de palabras**: 1–2-gramas, min_df=5, max_df=0.5 o 0.95 según etapa, sublinear_tf=True cuando aplica.
  - **TF-IDF de caracteres**: 3–5-gramas, min_df=5, max_df=0.95.
  - **Fusión**: concatenación dispersa [TF-IDF palabras | TF-IDF caracteres].
- **Co-ocurrencias y PPMI**
  - Vocabulario top-V=10 000 y ventana simétrica ±4.
  - Matriz COO simétrica de co-ocurrencias.
  - **PPMI**: ((i,j)=_2), (=(,0)).
- **Embeddings Word2Vec**
  - *Skip-gram*, vector_size=100, window=5, min_count=5, negative=10, epochs=5.
  - Vector de documento: promedio de vectores de palabras presentes.
  - Visualización exploratoria con PCA y t-SNE sobre subconjuntos.

3.7. Modelado probabilístico de lenguaje

- Construcción de unigramas, bigramas y trigramas con marcadores <s>, </s> y <unk> para rareza.
- **Suavizado add-k** en bi/tri-gramas (k=0.1 por defecto).
- **Kneser-Ney** para bigramas con descuento D=0.75 y probabilidad de continuación.
- Cálculo de entropía (H) y perplejidad (PP=2^H) a nivel de corpus y de oración.

3.8. Clasificación supervisada

- Variable objetivo: sentimiento ∈ {0,1}.
- **Partición**: train_test_split(test_size=0.20, stratify=y, random_state=42).
- **Representación final**: concatenación de TF-IDF de palabras (1–2-gramas) y de caracteres (3–5-gramas).
- **Modelos**:
    - LinearSVC(C=1.0, random_state=42
    - MultinomialNB(alpha=0.5)
- **Métricas**: *accuracy*, *macro-F1*, *classification_report* por clase, y **matriz de confusión** para el mejor modelo.

3.9. Reproducibilidad y consideraciones prácticas

- Semillas: random_state=42 y random.seed(0) donde aplica.
- Control de esparsidad y memoria mediante min_df y límite de vocabulario.
- Visualizaciones con Matplotlib; reducción con PCA/t-SNE de *sklearn*.

4. Resultados

4.1. Estadísticas de representación

BoW / TF-IDF

- Documentos: **49 599**
- Términos distintos: **151 860**
- Densidad promedio: **0.000777**
- Entradas no nulas (nnz): **5 854 820**
- Promedio de términos no nulos por documento: **118.0**

Co-ocurrencias y PPMI

- Dimensión: **(10 000, 10 000)**
- nnz: **11 637 960**
- Densidad: **0.11637960**
- Total de co-ocurrencias: **65 419 004**
- Media de conteo por par: **5.62**
- PPMI medio (>0): **2.0399**
- PPMI máximo: **12.4341**

Ejemplos de normalización

- “Uno de los otros críticos…” → “uno de los otros criticos…”
- “Una pequeña pequeña producción…” → “una pequena pequena produccion…”
- “Pensé que…” → “pense que…”

Stopwords

- ES base: **313** | ES extra: **608**
- EN base: **198** | EN extra: **1 298**
- Finales: **1 844**
- Ceros tras limpieza de stopwords: **0**

4.2. Modelado probabilístico

Modelo	k/D	H (bits/token)	PP
Bigram Add-k	0.1	13.450	11 191.71
Trigram Add-k	0.1	15.849	59 040.98
Bigram Kneser-Ney	0.75	12.642	6 389.73

Tabla 1: N-gramas

Ejemplos de oraciones:

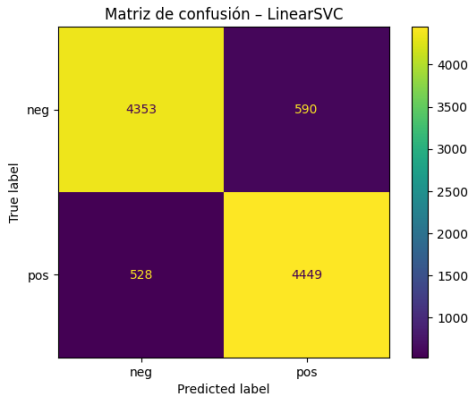
- *la pelicula fue excelente*
  - Bigram Add-k → H = 7.458 bits, PP = 175.80
  - Trigram Add-k → H = 11.322 bits, PP = 2 559.64
  - Bigram KN → H = 7.799 bits, PP = 222.76
- *la trama resulto aburrida*
  - Bigram Add-k → H = 10.409 bits, PP = 1 359.84
  - Trigram Add-k → H = 16.310 bits, PP = 81 252.76
  - Bigram KN → H = 10.115 bits, PP = 1 109.24

4.3. Clasificación supervisada

Modelo	Representación	Accuracy	F1
LinearSVC	TF-IDF palabras + caracteres	0.8873	0.8884
MultinomialNB	TF-IDF palabras + caracteres	0.8550	0.8558

Tabla 2: Comparación de métricas entre modelos

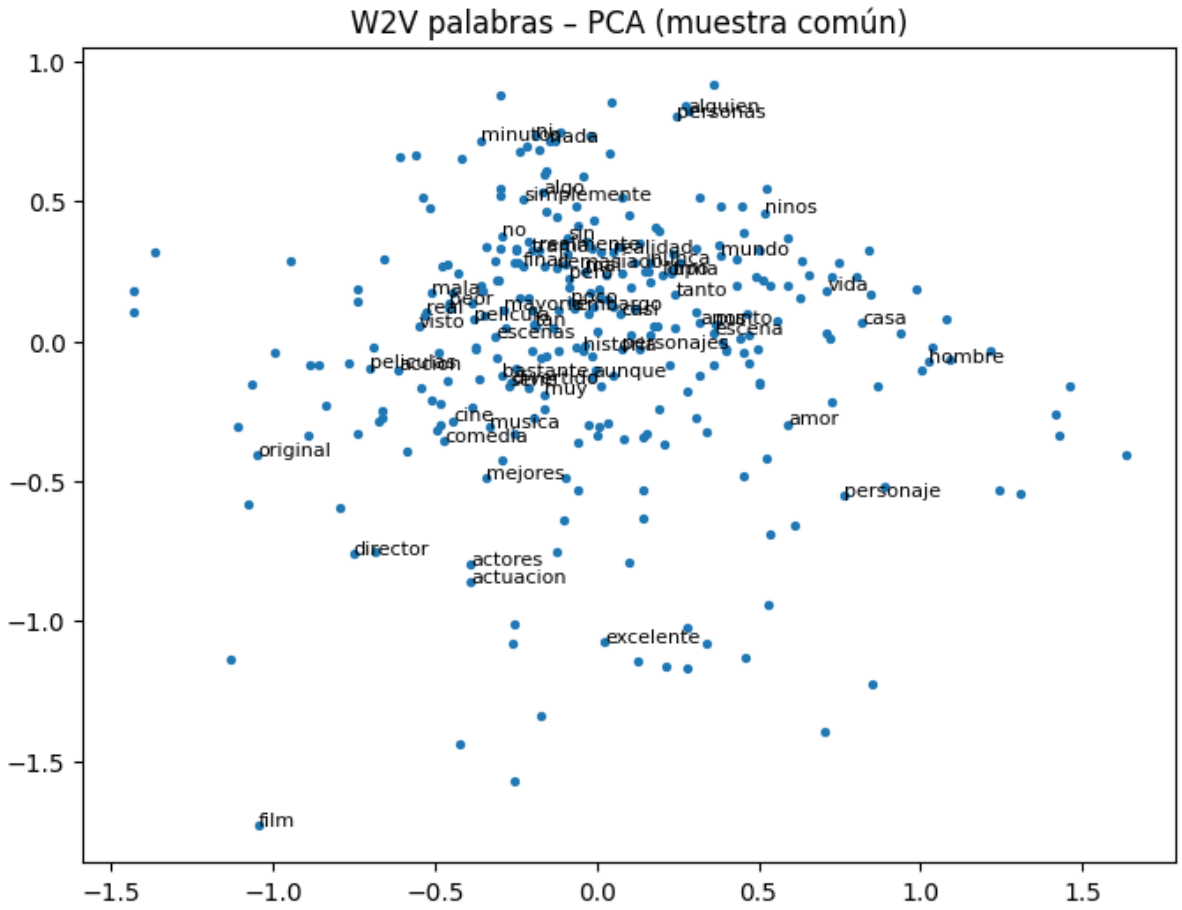
Mejor modelo: LinearSVC basado en F1.

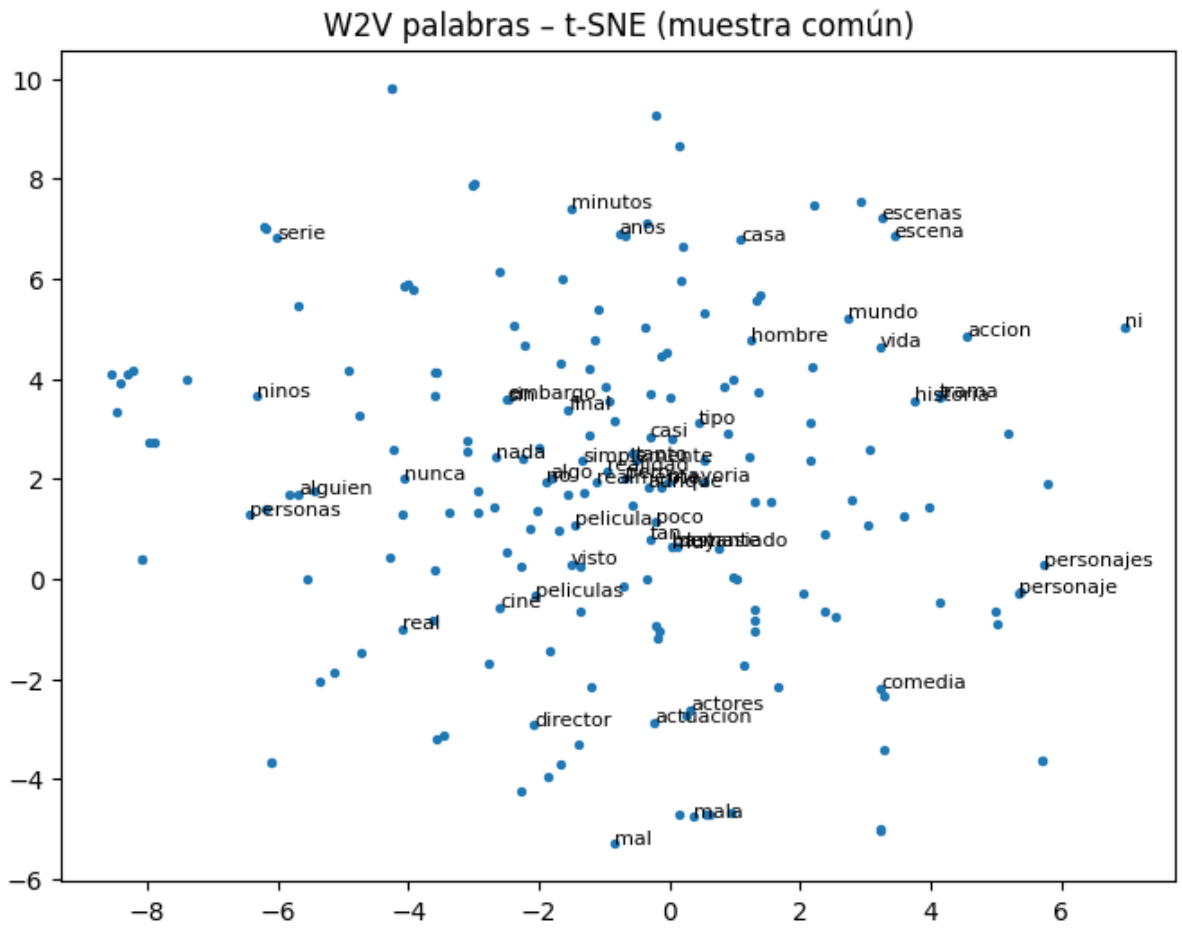


4.4. Visualizaciones semánticas

Word2Vec

- Vocabulario: **51 380**
- Dimensión: **100**
- Palabras similares (muestra):
  - *excelente* → {excepcional, sobresaliente, fantastica, maravilloso, brillante}
  - *terrible* → {horrible, atroz, horrenda, atroçada, execrable}
  - *divertido* → {entretenido, humoristico, gracioso, hilarante, sketismo}
  - *aburrido* → {tedioso, inimaginativo, hackneado, rutinario, repetitivo}




W2V palabras – t-SNE (muestra común)
minutos
escenas
serie
años
escena
casa
mundo
ni
hombre
vida
accion
ninos
historia
tipo
casi
nada
nunca
alguien
personas
pelicula
poco
visto
personajes
personaje
peliculas
cine
real
comedia
actores
director
actuación
mala
mal

5. Discusión crítica

Fortalezas

- Limpieza robusta y protección de negaciones e intensificadores.
- Uso combinado de n-gramas de palabras y de caracteres mejora cobertura OOV.
- Línea base sólida con SVM y NB.

Limitaciones

- Sin validación cruzada ni *tuning* sistemático de hiperparámetros.
- *Stemming* puede degradar matices semánticos; no se ensayó lematización.
- Corrección ortográfica no integrada al *pipeline*.
- PPMI y co-ocurrencias son costosos en memoria y no alimentan al clasificador.
- Posible sesgo del dataset y ruido de traducción.

Mejoras propuestas

- Validación cruzada estratificada y búsqueda de hiperparámetros (p. ej., `C` en SVM, `alpha` en NB, `min_df`, `ngram_range`).
- Sustituir *stemming* por lematización en español y tokenización con spaCy.
- Incorporar **FastText** o modelos **transformers** en español (p. ej., BETO) con *fine-tuning* y *early stopping*.
- Análisis de errores por longitud, polaridad mixta y negación; ajuste de umbrales y calibración.sin
- Interpretabilidad: pesos de SVM, n-gramas más informativos, y SHAP sobre versiones densas.
- *Pipelines* reproducibles con `Pipeline`/`ColumnTransformer` y exportación con `joblib`.

6. Conclusiones

El pipeline clásico con TF-IDF de palabras y caracteres logró un rendimiento competitivo ($F1 = 0.8884$) usando LinearSVC, superando a Naive Bayes. La limpieza, normalización y protección de negaciones fueron claves para mantener coherencia lingüística.

Los modelos probabilísticos mostraron que Kneser-Ney reduce entropía y perplejidad frente a add-k, aunque los trigramas incrementan la sparsidad. Las visualizaciones de Word2Vec revelaron agrupamientos semánticos claros entre términos positivos y negativos, confirmando la capacidad del modelo para capturar relaciones léxicas.