Universidad Francisco de Vitoria

Escuela Politécnica Superior

Deep Learning

Práctica II:
Natural Language Processing

Profesor:
D. Moises Martínez
Alumnos:
Alfredo Robledano
Rubén Sierra
Jorge Barcenilla
Pedro García

10 de mayo de 2025

Índice

1. Resumen

Esta práctica explora cómo el tamaño de los vectores de embeddings influye en la identificación contextual de palabras, utilizando redes neuronales profundas. Los embeddings, como representaciones vectoriales densas, permiten capturar relaciones semánticas y contextuales entre términos, lo que resulta fundamental para tareas de procesamiento del lenguaje natural como la clasificación de texto o el análisis de sentimientos. En particular, las redes neuronales desarrolladas en este trabajo se aplican al análisis de los sentimientos expresados en comentarios de redes sociales, permitiendo identificar de manera automática la polaridad emocional -positiva, negativa o neutra- de los mensajes publicados por los usuarios. De este modo, se evalúa cómo la dimensionalidad de los embeddings afecta la capacidad de los modelos para comprender e interpretar el contenido emocional de los textos

2. Desarrollo

2.1. Preparación del *corpus* y preprocesamiento

Optamos por subir el *dataset* a Google Drive, una decisión motivada por la necesidad de contar con una ubicación centralizada y accesible desde entornos de computación en la nube, como Google Colab. Esta elección facilitó la integración del *dataset* en el *pipeline* de trabajo, al permitir su descarga automática mediante `gdown`, una herramienta que posibilita la descarga directa desde Google Drive al entorno de ejecución.

Los datos, estructurados en archivos CSV (`train.csv` y `test.csv`), contenían metadatos demográficos y temporales que enriquecieron el contexto semántico de los textos, como el país de origen, la edad del usuario y el sentimiento del *tweet*. Este último resultó fundamental para la tarea de clasificación, aunque los demás no pudieron ser aprovechados en este trabajo.

El proceso de extracción empleó `ThreadPoolExecutor` para descomprimir los archivos en paralelo mediante hilos de forma optimizada. Se eliminó el directorio `__MACOSX` con el fin de garantizar la integridad del *dataset*, eliminando residuos generados por sistemas operativos ajenos al entorno experimental, en este caso, macOS.

Durante la carga de los archivos, se utilizó la codificación `ISO-8859-1`, ya que con `UTF-8` se presentaban caracteres no reconocidos, lo que impedía una correcta lectura de los datos.

Para el tratamiento lingüístico, se implementó un proceso de normalización del texto que consistió en la eliminación de caracteres especiales mediante expresiones regulares, conversión a minúsculas para homogeneizar el léxico y tokenización utilizando la clase `Tokenizer` de `Keras`

Se utilizó la técnica de *skip-grams* debido a su capacidad para capturar relaciones contextuales entre palabras. Este enfoque permitió generar pares de palabras formados por una palabra objetivo y su contexto, logrando un equilibrio entre la detección de patrones y la distribución del vocabulario. Hemos implementado una función propia de *skip-grams*, ya que la versión de `Keras` se encuentra deprecada y no reflejaba fielmente el contexto real de cada oración, dado que introducía índices aleatorios dentro del tamaño del vocabulario al pasarle una única oración del *dataset*.

```
target: were | context: have
target: were | context: responded
target: were | context: if
target: were | context: i
target: were | context: going
target: going | context: responded
target: going | context: if
target: going | context: i
target: going | context: were
tf.Tensor(5, shape=(), dtype=int64) 4518
tf.Tensor(8, shape=(), dtype=int64) 5577
tf.Tensor(6, shape=(), dtype=int64) 7275
tf.Tensor(3, shape=(), dtype=int64) 9695
tf.Tensor(3, shape=(), dtype=int64) 814
tf.Tensor(8, shape=(), dtype=int64) 8378
tf.Tensor(3, shape=(), dtype=int64) 2367
tf.Tensor(8, shape=(), dtype=int64) 5704
tf.Tensor(6, shape=(), dtype=int64) 907
tf.Tensor(6, shape=(), dtype=int64) 1619
```

Figura 1: Ejemplo del uso de *skip-grams* de `Keras` que muestra índices aleatorios

2.2. Arquitectura del modelo

3. Conclusión

En conclusión, el trabajo realizado ha permitido profundizar en la aplicación de técnicas de procesamiento de lenguaje natural basadas en redes neuronales profundas para el análisis de sentimientos. A través de una cuidadosa preparación y preprocesamiento del corpus, así como la implementación de una función personalizada de skip-grams, se ha logrado construir un pipeline robusto y eficiente para la generación de embeddings semánticos. Estos embeddings, entrenados específicamente sobre el dominio de los textos analizados, han demostrado ser capaces de capturar tanto relaciones contextuales locales como patrones globales en el lenguaje, lo que resulta fundamental para tareas de clasificación de sentimientos. El enfoque adoptado ha evidenciado la importancia de adaptar y personalizar las herramientas y técnicas existentes a las particularidades del corpus y los objetivos del proyecto, como las dimensiones de los embeddings y la arquitectura de la red neuronal.