1. Descripción del corpus para NER y sus principales características

El corpus Seleccionado para realizar el objetivo a es Amazon Fine Food Reviews, la cual fue tomada desde Kaggle(https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews/data)

Este corpus contene reseñas de productos alimenticios de Amazon. El corpus completo contiene más de 500,000 reseñas. Y esta en un formato: CSV con columnas como Text, Score, UserId, ProfileName, Time, Summary, Helpful.

Este corpus se utilizará para extracción de entidades relevantes (Named Entities).

Los datos incluyen:

Reseñas de octubre de 1999 a octubre de 2012

568,454 opiniones

256.059 usuarios

74.258 productos

260 usuarios con > 50 reseñas

2. Descripción de la técnica de KeyPhrase Extraction

El objetivo es extraer palabras clave relevantes del texto que puedan indicar los aspectos más importantes mencionados en las reseñas. Para esto se realizó el siguiente proceso:

- **Tokenización:** Dividir el texto en palabras (tokens) individuales.
- **Eliminación de Stop Words**: Eliminar palabras comunes que no aportan significado relevante.
- **Lematización:** Convertir las palabras a su forma base o lema.
- **Etiquetado PoS (Part of Speech):** Etiquetar cada token con su categoría gramatical correspondiente (sustantivo, adjetivo, etc.).

3. Definición detallada de la tarea de extracción de NER, tipos de etiqueta, etc.

Para identificar entidades relevantes en este caso, **atributos de productos** el texto, nos dio un resultado mas exacto que al probar con otras características.

El objetivo es identificar y etiquetar atributos específicos de los productos mencionados en las reseñas, como "sabor", "precio", "calidad", etc.

Tipos de Etiqueta, utilizamos etiquetas PoS para identificar sustantivos y adjetivos relevantes que describen los atributos de los productos.

Utilizamos una lista predefinida de atributos relevantes y expresiones regulares para buscar estos atributos en las oraciones.

4. Exploratoria estadística elemental del corpus para NER

Trabajamos con una muestra de 5000 reseñas. Observamos términos frecuentes relacionados con atributos de productos como "taste", "price", "quality", "flavor", etc.

Identificamos que muchos usuarios mencionan atributos específicos de los productos en sus reseñas, proporcionando una buena base para la extracción de información.

5. Para el caso de la tarea de fine-tuning

Descripción del Modelo Seleccionado

El modelo seleccionados es DistilBERT (Distilled BERT)

Características principales:

Arquitectura: DistilBERT es una versión más pequeña y rápida de BERT (Bidirectional Encoder Representations from Transformers), manteniendo el 97% del rendimiento de BERT mientras es 60% más pequeño y dos veces más rápido.

Capas: DistilBERT tiene 6 capas transformadoras, en comparación con las 12 capas de BERT base.

Parámetros: Aproximadamente 66 millones de parámetros, frente a los 110 millones de parámetros del BERT base.

Entrenamiento: Fue entrenado utilizando el proceso de "knowledge distillation", donde el modelo más pequeño (DistilBERT) aprende a imitar el comportamiento de un modelo más grande (BERT) durante el entrenamiento.

Tareas: Se puede aplicar a una variedad de tareas de procesamiento de lenguaje natural (NLP) como clasificación de texto, respuesta a preguntas, y más.

Descripción del Corpus Seleccionado

Corpus utilizado es **mteb/mtop_domain** tomado de mteb/mtop_domain · Datasets at Hugging Face

Características principales:

El corpus está enfocado en la clasificación de dominio. Los dominios pueden incluir áreas temáticas como viajes, restaurantes, clima, noticias, deportes, entre otros.

Contiene múltiples etiquetas que representan diferentes dominios. Verificamos que hay múltiples etiquetas únicas en el dataset, y el número de etiquetas se ajusta en el modelo en consecuencia.

Cada ejemplo en el corpus consiste en un texto y una etiqueta correspondiente al dominio del texto.

Explicación de las Decisiones de Tokenización, Padding y Parametrización

Tokenización

El tokenizer de DistilBERT está optimizado para la arquitectura del modelo y ya ha sido preentrenado en un corpus grande, lo que facilita una tokenización eficiente y precisa.

Padding

Aplicar padding con max_length. El padding asegura que todas las secuencias de entrada tengan la misma longitud, lo cual es necesario para el procesamiento en lotes y eficiente

del modelo. Usar max_length asegura que todas las secuencias tengan una longitud uniforme.

Parametrización

Configurar los parámetros de entrenamiento del modelo.Elegir parámetros óptimos para el entrenamiento del modelo basado en las mejores prácticas y la naturaleza del dataset.

Learning Rate (Tasa de Aprendizaje): 2e-5, una tasa comúnmente usada que proporciona un buen equilibrio entre velocidad de convergencia y estabilidad.

Batch Size: 16 para tanto el entrenamiento como la evaluación, lo que permite un balance entre el uso de memoria y la eficiencia computacional.

Número de Épocas: 3, un número típico que permite al modelo aprender adecuadamente sin sobreajustarse.

Weight Decay: 0.01, utilizado para regularización y evitar el sobreajuste.

6. Explicación detallada, justificación de librerías y detalle de resultados para el proceso de limpieza y preparación de los datos.

El código presentado realiza el preprocesamiento de texto utilizando las bibliotecas spaCy y pandas.

Descripción de las Librerías

spaCy: Biblioteca de NLP utilizada para tokenizar texto, eliminar stop words y lematizar palabras. Es conocida por su velocidad y precisión.

pandas: Biblioteca de manipulación de datos utilizada para gestionar y transformar el conjunto de datos estructurados en DataFrames.

Justificación

spaCy es rápido y eficiente, ideal para grandes volúmenes de datos.ofrece herramientas integradas que facilitan el preprocesamiento. Además, pandas se integra bien con spaCy, facilitando la manipulación y análisis de datos.

Resultados Obtenidos

El texto original se transforma a una forma más limpia y uniforme. se ha convertido a minúsculas, y Eliminación de stop words y puntuación.

7. Explicación detallada, justificación de librerías y detalle de resultados para el proceso de Extracción de palabras clave (KPE) y PoS tagging con librerías pre-entrenadas.

Librerías Utilizadas:

spaCy se utiliza para cargar un modelo preentrenado de inglés (en_core_web_sm), tokenizar el texto, identificar partes del discurso, y extraer entidades nombradas.

Pandas Se utiliza para almacenar y manipular el conjunto de datos, en este caso, un DataFrame con una columna de texto que se va a preprocesar y analizar.

Algoritmos y Técnicas Utilizadas:

PoS Tagging (Etiquetado de Partes del Discurso), Es el proceso de asignar etiquetas a cada palabra en un texto, indicando su parte del discurso, como sustantivo (NOUN), adjetivo (ADJ), verbo (VERB), etc.

spaCy etiqueta las palabras en el texto, permitiendo la identificación de sustantivos y adjetivos para la extracción de palabras clave.

NER (Reconocimiento de Entidades Nombradas),Es el proceso de identificar y clasificar entidades nombradas en un texto, como nombres de personas, organizaciones, ubicaciones, etc. identifica y clasifica las entidades nombradas en el texto preprocesado.

Justificación en la Selección de las Librerías y Algoritmos

spaCy:

Proporciona modelos preentrenados altamente precisos para diversas tareas de NLP. Ofrece una amplia gama de funcionalidades integradas que simplifican tanto el etiquetado de partes del discurso como la extracción de entidades nombradas.

Detalle de los Resultados Obtenidos

Después de aplicar el proceso de extracción de palabras clave y entidades nombradas, se obtiene un nuevo DataFrame con una columna adicional Keywords_Entities que contiene una lista de palabras clave (sustantivos y adjetivos) y entidades nombradas del texto preprocesado. Los resultados muestran palabras Clave: Sustantivos y adjetivos que son significativos en el contexto del texto.

8. Explicación detallada, justificación de librerías y detalle de resultados para el proceso de Reconocimiento de entidades relevantes (NER) por medio de heurísticas, regexp, etiquetas PoS, lexicones, etc.

Librerías Utilizadas:

spaCy Se utiliza para tokenizar el texto y analizarlo, lo que permite la identificación de sustantivos y adjetivos.

re (expresiones regulares), es un módulo de Python que proporciona herramientas para trabajar con expresiones regulares. Las expresiones regulares permiten buscar y manipular cadenas de texto utilizando patrones específicos.

Se utiliza para definir un patrón que busca coincidencias con una lista predefinida de atributos de productos.

Algoritmos y Técnicas Utilizadas:

Expresiones Regulares (Regexp), Las expresiones regulares son secuencias de caracteres que forman un patrón de búsqueda. Se utilizan para buscar coincidencias específicas dentro de cadenas de texto. Se define un patrón que busca términos relacionados con atributos de productos en el texto.

Heurísticas, son métodos basados en la experiencia y reglas prácticas que ayudan a resolver problemas de manera eficiente. Se utilizan heurísticas para identificar frases que probablemente contengan atributos de productos, basándose en patrones comunes y etiquetas de partes del discurso.

Justificación en la Selección de las Librerías y Algoritmos

re (expresiones regulares), Las expresiones regulares permiten definir patrones de búsqueda muy precisos y flexibles, adecuados para identificar términos específicos en el texto.

Detalle de los Resultados Obtenidos

Después de aplicar el proceso de identificación de atributos de productos, se obtiene un nuevo DataFrame con una columna adicional Product_Attributes que contiene una lista de atributos de productos encontrados en el texto preprocesado. Los resultados muestran:

Atributos de Productos Identificados: Palabras que coinciden con los términos en la lista predefinida de atributos, como "taste", "price", "quality", etc.

9. Explicación detallada, justificación de librerías y detalle de resultados para el proceso de Reconocimiento de entidades relevantes (NER) por medio de un modelo pre-entrenado.

Librerías Utilizadas:

Transformers de HuggingFace, Transformers es una biblioteca de código abierto de HuggingFace que proporciona herramientas para trabajar con modelos de lenguaje preentrenados, como BERT, GPT-2, y muchos otros. Incluye pipelines que facilitan la

implementación de tareas de NLP. Se utiliza el pipeline de reconocimiento de entidades nombradas (NER) con el modelo preentrenado dslim/bert-base-NER.

Algoritmos y Técnicas Utilizadas:

NER (Reconocimiento de Entidades Nombradas), Es el proceso de identificar y clasificar entidades nombradas en un texto, tales como nombres de personas, organizaciones, ubicaciones, entre otros. El pipeline NER de HuggingFace analiza el texto y devuelve las entidades nombradas junto con sus etiquetas.

Justificación en la Selección de las Librerías y Algoritmos

Transformers de HuggingFace, Los modelos preentrenados en Transformers, como BERT, han demostrado un rendimiento sobresaliente en diversas tareas de NLP, incluido NER.

Los pipelines de HuggingFace simplifican la implementación de tareas complejas de NLP, permitiendo a los desarrolladores aplicar modelos avanzados con pocas líneas de código. HuggingFace ofrece una amplia gama de modelos preentrenados que pueden adaptarse a diferentes necesidades y lenguajes.

Descripción de los Problemas Encontrados y las Soluciones Adoptadas para el objetivo 1

- Inicialmente, la identificación de nombres de productos no fue precisa. Por lo que cambiamos el enfoque a la identificación de atributos específicos de los productos.
- Tiempo de entrenamiento elevado debido al tamaño del modelo. Por lo que se redujo el tamaño del conjunto de datos y se ajustaron los hiperparámetros para un entrenamiento más rápido.

Descripción de los Problemas Encontrados y las Soluciones Adoptadas para el objetivo 2

Durante el proceso de fine-tuning del modelo DistilBERT utilizando el dataset mteb/mtop_domain, se encontraron varios problemas. A continuación se describen estos problemas y las soluciones adoptadas para resolverlos.

Hubo Conflicto de Dependencias en Google Colab, en primera instancia estaba realizando el ejercicio en jupyter notebook pero tuve inconvenientes con las instalaciones de las librerías y el tiempo de proceso.

Al intentar instalar las bibliotecas necesarias (transformers, datasets, accelerate), surgió un conflicto de dependencias con la versión de requests requerida por Google Colab.

Configuración de Tokenización y Padding, Se necesitaba asegurar que todas las secuencias de entrada tuvieran una longitud uniforme para el procesamiento eficiente del modelo.

Solución Adoptada:

Se instalaron las versiones específicas de las bibliotecas necesarias para resolver el conflicto y se decidió realizar el ejercicio en Google colab con GPU.

Se aplicó padding con max_length y truncamiento para asegurar que todas las secuencias tuvieran la misma longitud.