Proyecto Final

Tratamiento de Datos
Máster de Ing. de Telecomunicación

Octubre 2024

1 Introducción

En este proyecto, los alumnos harán uso de los conocimientos y técnicas adquiridos durante el curso para resolver una tarea de aprendizaje sobre documentos textuales. Los alumnos trabajarán individualmente o por parejas sobre documentos que recogen recetas de cocina del mundo.

Las tareas a realizar incluirán necesariamente:

- Procesado y homogeneización de textos.
- Representación vectorial de los documentos con las siguientes técnicas: TFIDF, Word2Vec, embeddings contextuales basados en Transformers.
- Resolución de una tarea de regresión.

El proyecto tiene una **valoración máxima de 3 puntos**, consistentes en:

- **Proyecto básico:** 2,25 puntos
- **Extensión:** 0,75 puntos

La fecha límite de entrega será el 13 de diciembre del 2024, a las 23,59 horas.

A continuación se indican los requisitos de cada una de las partes.

2 Descripción del conjunto de datos

El conjunto de datos se proporciona en el fichero JSON *"full_format_recipes.json"*[1]. Dicho dataset consta de 20 130 entradas correspondientes a recetas de la web `www.epicurious.com`. Para cada receta se proporciona la siguiente información:

- *directions*: Instrucciones para hacer la receta.

[1] Recuerde que puede leer un fichero JSON como un `DataFrame` de Pandas con la función `pd.read_json(path_file)`.Puede consultar más detalles en la documentación de Pandas.

- *categories*: Distintas categorías que se asignan al plato.
- *desc*: Descripción.
- *title*: Título de la publicación.
- *rating*: Puntuación dada por usuarios.
- *fat*: Cantidad de grasa en gramos.
- *protein*: Cantidad de proteínas en gramos.
- *calories*: Cantidad de calorías en gramos.
- *sodium*: Cantidad de sodio en gramos.
- *ingredients*: Cantidad de cada ingrediente.
- *date*: Fecha de publicación.

Las variables *directions*, *desc*, *categories* y *title* contienen texto y el resto contienen valores numéricos. Se utilizará la variable *rating* como **salida** para solucionar un problema de **regresión**

3 Proyecto básico

El proyecto básico consistirá en la resolución de una tarea de regresión, comparando las prestaciones obtenidas al utilizar distintas vectorizaciones de los documentos y al menos dos estrategias distintas de aprendizaje automático, según se describe a continuación. Los pasos que debe seguir en su trabajo son los siguientes:

1. Análisis de variables de entrada. Visualice la relación entre la variable de salida y algunas de las categorías en la variable *categories* y explique su potencial relevancia en el problema.
2. Implementación de un pipeline para el preprocesado de los textos. Para esta tarea puede usar las librerías habituales (NLTK, Gensim o SpaCy), o cualquier otra librería que considere oportuna. Tenga en cuenta que para trabajar con transformers el texto se pasa sin preprocesar.
3. Representación vectorial de los documentos mediante tres procedimientos diferentes:
    - TF-IDF
    - Word2Vec(es decir, la representación de los documentos como promedio de los embeddings de las palabras que lo forman)
    - Embeddings contextuales calculados a partir de modelos basados en transformers (e.g., BERT, RoBERTa, etc).

4. Entrenamiento y **evaluación** de modelos de regresión utilizando al menos las dos estrategias siguientes de aprendizaje automático:
    - Redes neuronales utilizando PyTorch para su implementación.
    - Al menos otra técnica implementada en la librería Scikit-learn (e.g., K-NN, SVM, Random Forest, etc)
5. Comparación de lo obtenido en el paso 3 con el fine-tuning de un modelo preentrenado con *Hugging Face*. En este paso se pide utilizar un modelo de tipo transformer con una cabeza dedicada a la tarea de regresión.

Se deberá utilizar la información en la variable *directions* y/o *desc* para todos los pasos del proyecto, pudiéndose combinar la información de estas variables con alguno de los metadatos en las otras variables. Deberá utilizar métricas para la evaluación adecuadas para los problemas de regresión. Las prestaciones de los distintos métodos deben estimarse con alguna metodología de validación que también deberá explicar en la documentación. Deberá aportar una descripción de la metodología empleada y analizar las prestaciones obtenidas según las variables de entrada.

Tenga en cuenta que el objetivo es describir el trabajo realizado y hacer un análisis crítico de los resultados obtenidos. Apóyese para ello en gráficas u otras representaciones que considere oportunas. No es necesario describir los algoritmos utilizados, aunque sí deberá explicar cómo ha realizado el ajuste de sus parámetros.

4 Extensión

El trabajo de extensión es libre, deberá ampliar el proyecto básico en la dirección que considere más oportuna. Por ejemplo:

- Uso de un *summarizer* preentrenado (utilizando pipelines de Hugging Face) para proporcionar un resumen de la variable *directions*, la cual es una lista de instrucciones que puede contener textos relativamente grandes así como pasos repetidos.
- Estudiar la capacidad de los modelos de tipo transformer para la generación de nuevas recetas. Aquí también se pueden comparar las prestaciones de esto respecto a su implementación con técnicas de prompting sobre modelos del lenguaje de uso libre (LLAMa, Mixtral, etc.).
- Explorar el potencial de técnicas de NLP como el uso de bigramas, part-of-speech tagging, tesauros, etc., (explotando, por ejemplo, la funcionalidad disponible en la librería *NLTK* de Python).
- Comparación de prestaciones utilizando distintos embeddings contextuales.
- Visualización y análisis empleando técnicas basadas en grafos.

Tome esta lista como una mera sugerencia, puede elegir cualquier otro tema siempre que encaje dentro del ámbito de la asignatura.

En el trabajo de extensión se valorará la creatividad y originalidad en la elección. Si tiene dudas sobre la idoneidad de la extensión elegida, consulte con el profesor. Tenga en cuenta, en todo caso, que el trabajo de extensión constituye 0,75 puntos sobre la nota final. Evite embarcarse en trabajos de extensión demasiado ambiciosos que puedan comprometer la entrega en plazo del proyecto.

Si tiene cualquier duda sobre la idoneidad de cualquier trabajo de extensión, consulte con el profesor.

5 Entrega del Proyecto

Para realizar la entrega del proyecto se utilizará **GitHub** como herramienta de colaboración y control de versiones. La entrega deberá incluir los siguientes elementos:

1. **Repositorio en GitHub**:
   - Deberán tener actualizado el repositorio de GitHub con el código utilizado durante el desarrollo del proyecto. El repositorio debe estar bien estructurado y contener toda la información necesaria para la ejecución del proyecto.
   - El repositorio debe incluir:
     - Código fuente debidamente comentado.
     - Los scripts necesarios para la preparación de datos y entrenamiento de los modelos.
     - Los notebooks de experimentación (si aplica) o scripts de análisis.
2. **Documentación**:
   - El repositorio debe contener un archivo `README.md` que describa claramente el proyecto, la metodología utilizada, las decisiones de implementación y la descripción de los resultados.
   - Alternativamente o adicionalmente, los estudiantes pueden utilizar **GitHub Pages** para crear una página web donde expliquen de forma más detallada el trabajo realizado, los resultados obtenidos, y cualquier análisis o conclusión relevante.

   La documentación deberá incluir:
   - Descripción del problema y objetivos del proyecto.
   - Metodología aplicada (procesamiento de los datos, modelos utilizados, etc.).
   - Análisis de resultados: incluir gráficos, métricas y una discusión sobre el rendimiento de los modelos.

3. **Link al repositorio**:

   - El repositorio debe estar público.
   - La entrega se hará a través de Aula Global: un estudiante de cada grupo deberá subir el enlace al repositorio compartido.
   - No se podrán editar las páginas de documentación después de la fecha de entrega.
   - Se evaluará el último commit antes del deadline.

Inexcusablemente, la documentación debe respetar el principio de reconocimiento de autorías. Si utilizan fragmentos de código ajenos o cualquier material procedente de fuentes externas, debe especificarlo claramente en la memoria.

6 Evaluación

El proyecto se evaluará considerando la documentación entregada así como con una presentación individual que se realizará la semana del 16 de diciembre. La evaluación se hará de acuerdo con los criterios siguientes:

- Proyecto básico (2,25 puntos)
  - Metodología (0,75)
  - Calidad de la documentación (0,3)
  - Calidad del código (0,3)
  - Calidad de la presentación (0,5)
  - Respuestas a comentarios de la presentación (0,4)
- Extensión (0,75 puntos)
  - Originalidad (0,25)
  - Calidad del trabajo (0,5)

A Información complementaria

En este apéndice disponen de información que puede resultar útil y deben tener clara para resolver el problema.

A.1 Extracción de Embeddings con un Transformer

En esta sección, explicaremos cómo extraer embeddings a partir de un modelo de transformer, específicamente utilizando BERT como ejemplo. Esta técnica es fundamental para obtener representaciones vectoriales de palabras o frases en el contexto del problema de NLP que se desea resolver.

Paso 1: Cargar el modelo preentrenado

El primer paso es cargar un modelo preentrenado de BERT desde la librería `transformers` de Hugging Face, que nos permitirá obtener los embeddings directamente. Puedes hacerlo de la siguiente manera:

```
from transformers import BertModel, BertTokenizer
import torch

# Cargar el tokenizer y el modelo preentrenado de BERT
tokenizer =
↪  BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Tokenizar una frase de ejemplo
text = "Ejemplo de extracción de embeddings con BERT"
input_ids = tokenizer.encode(text, return_tensors='pt')
```

Paso 2: Obtener los embeddings

Con el modelo y la frase tokenizada, podemos pasar la entrada al modelo para obtener los embeddings. BERT devolverá dos salidas: los embeddings de la capa oculta y los embeddings del token `[CLS]` (que puede ser usado para tareas de clasificación).

```
        # Obtener los embeddings
        with torch.no_grad():
            outputs = model(input_ids)

        # Extraer los embeddings de la última capa oculta
        last_hidden_states = outputs.last_hidden_state
```

El tensor `last_hidden_state` contiene los embeddings de todos los tokens de la frase, y puedes usarlos según lo que necesites (por ejemplo, promediarlos para obtener una representación de la frase).

Más información

Para una explicación más detallada sobre cómo funcionan los embeddings en BERT, así como otros detalles sobre su utilización, puedes consultar el siguiente tutorial disponible en este enlace:
https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/#why-bert-embeddings.

A.2 Fine-Tuning de un Modelo Preentrenado de Hugging Face

El fine-tuning consiste en ajustar un modelo preentrenado como BERT para una tarea específica, como clasificación de texto o análisis de sentimientos. A continuación, explicamos los pasos básicos para hacer fine-tuning de un modelo preentrenado utilizando la librería `transformers` de Hugging Face.

Paso 1: Cargar el modelo y los datos

Primero, necesitamos cargar un modelo preentrenado y un tokenizer, además de preparar los datos de entrenamiento.

```
from transformers import BertTokenizer, BertForSequenceClassification
from transformers import Trainer, TrainingArguments

# Cargar el tokenizer y el modelo preentrenado de clasificacion
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)

# Tokenizar los datos de entrada
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=128)
test_encodings = tokenizer(test_texts, truncation=True, padding=True, max_length=128)
```

En este ejemplo, `BertForSequenceClassification` es el modelo de BERT ajustado para una tarea de clasificación con `num_labels=2` para clasificar en dos clases.

Paso 2: Crear el conjunto de datos

Después de tokenizar los textos, debemos crear un conjunto de datos compatible con el formato requerido por Hugging Face.

```
import torch

class Dataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val
        ↪  in self.encodings.items()}
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = Dataset(train_encodings, train_labels)
test_dataset = Dataset(test_encodings, test_labels)
```

Paso 3: Configurar el entrenamiento

Usamos la clase `Trainer` de Hugging Face para configurar y ejecutar el proceso de fine-tuning. Aquí configuramos los argumentos de entrenamiento, como el número de épocas y la tasa de aprendizaje.

```
    training_args = TrainingArguments(
        output_dir='./results',          # Directorio de salida
        num_train_epochs=3,              # Epocas
        per_device_train_batch_size=16,  # Batch entrenamiento
        per_device_eval_batch_size=64,   # Batch evaluacion
        warmup_steps=500,                # Steps de calentamiento
        weight_decay=0.01,               # Decaimiento del peso
        logging_dir='./logs',            # Directorio de logs
        logging_steps=10,
    )

    trainer = Trainer(
        model=model,                     # Modelo preentrenado
        args=training_args,              # Argumentos de entrenamiento
        train_dataset=train_dataset,     # Datos de entrenamiento
        eval_dataset=test_dataset        # Datos de evaluacion
    )

    # Iniciar el fine-tuning
    trainer.train()
```

Paso 4: Evaluación y guardado del modelo

Después de entrenar el modelo, podemos evaluarlo en el conjunto de datos de prueba y guardar el modelo ajustado.

```
        # Evaluar el modelo
        trainer.evaluate()

        # Guardar el modelo fine-tuned
        model.save_pretrained('./fine-tuned-bert')
        tokenizer.save_pretrained('./fine-tuned-bert')
```

Más información

Para más detalles sobre el fine-tuning y otros ejemplos de uso, puedes consultar la documentación oficial de Hugging Face en el siguiente enlace: https://huggingface.co/transformers/training.html

También puede consultar tutorial de Hugging Face para el fine-tuning que estará disponible en Aula Global.