TEMA 5 – HOJA DE EJERCICIOS III

Se plantean diferentes ejercicios basados en modelos preentrenados de Transformers para la obtención de embeddings contextualizados.

Ejercicio 1. Embeddings Contextualizados - BERT

A diferencia de los embeddings estáticos, que otorgan un único vector fijo a cada término independientemente del contexto en el que se encuentra, los embeddings contextualizados generan representaciones dinámicas que varían según el significado de los términos dentro de un texto concreto. Esta característica permite capturar mejor la ambigüedad y polisemia presentes en el lenguaje natural.

Uno de los modelos más representativos dentro de esta categoría es BERT. Este modelo de tipo de *encoder* utiliza una arquitectura basada en mecanismos de atención y Transformers, lo que le permite considerar el contexto completo en el que se encuentra un determinado token al generar su representación vectorial.

De este modo, BERT produce embeddings contextualizados a nivel de token en la salida de su última capa. Una vez obtenidas las salidas generadas por BERT, es posible acceder a las representaciones vectoriales de los tokens procesados mediante el atributo `last_hidden_state`:

```
outputs = model(**tokens)
outputs.last_hidden_state.numpy()
```

Dada la frase de consulta "Dogs are domestic animals." y el conjunto de frases ["Dogs are pets.", "This is a dog.", "They are free today."] se pide:

- Con el modelo y tokenizador de BERT 'bert-base-uncased', obtén el vector promedio *(mean pooling)* de la frase de consulta y el de cada frase del conjunto de frases. Nota que esta implementación de BERT utiliza los tokens [CLS], [SEP] y [PAD], por lo que se recomienda eliminar dichos tokens al hacer el vector promedio. Puedes utilizar la función del Tokenizer tokenizer.convert_ids_to_tokens(token_id) para localizar su posición exacta.
- Nuevamente, con el modelo y tokenizador de BERT 'bert-base-uncased', obtén el vector máximo *(max pooling)* de la frase de consulta y el de cada frase del conjunto de frases.

- Para las anteriores vectorizaciones, calcula la similitud coseno entre la frase de consulta y cada una del conjunto de frases. Reflexiona sobre qué frases son consideradas más similares por estos modelos y qué vectores funcionan mejor (media o máximo).

Nota:

- https://huggingface.co/google-bert/bert-base-uncased
- https://numpy.org/doc/2.2/reference/generated/numpy.mean.html
- https://numpy.org/doc/2.2/reference/generated/numpy.max.html

Ejercicio 2. Embeddings Contextualizados – SBERT

Recientemente, han surgido diversas alternativas basadas en la arquitectura original de BERT para el cálculo de embeddings contextuales. Una de estas alternativas es SBERT, adaptación que utiliza redes siamesas / bi-encoders específicamente fine-tuneados para tareas de recuperación de la información y comparación de textos cortos.

Nuevamente, dada la frase de consulta "Dogs are domestic animals." y el conjunto de frases ["Dogs are pets.", "This is a dog.", "They are free today."], se pide:

- Con el modelo de SBERT 'all-MiniLM-L6-v2', obtén los embeddings de la frase de consulta y de cada frase del conjunto de frases.
- Con el modelo de SBERT 'all-mpnet-base-v2', obtén los embeddings de la frase de consulta y de cada frase del conjunto de frases.
- Para los anteriores embeddings, calcula la similitud coseno entre la frase de consulta y cada una del conjunto de frases. Reflexiona sobre qué frases son consideradas más similares por estos modelos.
- El Massive Text Embedding Benchmark (MTEB) establece una comparativa sistemática entre diferentes propuestas de modelos de embeddings textuales en función de sus características y sus propósitos. Utilizando esta comparativa, explora otros modelos basados en SBERT para la comparación de las frases analizadas.

  Nota:

  - https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
  - https://huggingface.co/sentence-transformers/all-mpnet-base-v2
  - https://huggingface.co/spaces/mteb/leaderboard