Laboratorio: Word Embedings y Transformers para clasificación de texto

**Objetivos**

Con este laboratorio el alumno comparará diferentes modelos de clasificación de texto, todos mediante el uso de técnicas basadas en Word Embedings y Transformers. El alumno por tanto adquirirá dos competencias: primero, la capacidad de aplicar un modelo neuronal para la clasificación de texto. Segundo, la capacidad de comparar diferentes modelos entre sí.

**Descripción**

En esta actividad vamos a trabajar en clasificar textos. Se recorrerá todo el proceso desde traer el dataset hasta proceder a dicha clasificación. Durante la actividad se llevarán a cabo muchos procesos como la creación de un vocabulario, el uso de embeddings y la creación de modelos.

Las cuestiones presentes en esta actividad están basadas en un Notebook creado por François Chollet, uno de los creadores de Keras y autor del libro "Deep Learning with Python".

**Nota**: Esta actividad está generada en coordinación con la asignatura “Sistemas Cognitivos”. En concreto, en dicha asignatura se te propondrá realizar un ejercicio similar, pero con una aproximación tecnológica diferente.

Para realizar la actividad, se partirá del fichero “mia7\_lab.ipynb”. Dicho fichero contiene el código fuente en Python necesario para realizar las siguientes tareas:

1. Importar las librerías necesarias
2. Descargar los datos del dataset
3. Mezclar y separar los datos en catálogos de entrenamiento y validación
4. Tokenización del texto
5. (…)
6. Evaluación del clasificador

Además de dicho fichero, dispones de dos ficheros adicionales como son:

* “mia7\_lab\_red\_clasica.ipynb”.
* “mia7\_lab\_red\_transformers.ipynb”.

Estos ficheros completan el apartado (5) con una aproximación basada en redes clásicas y otra en transformers, respectivamente. En este laboratorio deberás cargar y ejecutar paso a paso las diferentes versiones del código.

Una vez ejecutado el código, responde a las siguientes preguntas:

1. Utilizando el tokenizador de spacy, que ya conoces, calcula el número promedio de tokens de una muestra de 15 ficheros de la categoría ‘com.graphics’. Indica el código utilizado y el resultado obtenido. **(1 punto)**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

1. El código proporcionado lee los ficheros uno a uno y, antes de generar el catálogo de datos de entrenamiento y validación, descarta las 10 primeras líneas de cada fichero. ¿Cuál es el trozo de código en el que se realiza dicho descarte?, ¿por qué crees que se descartan dichas líneas?, ¿por qué 10 y no otro número? **(1 punto)**

**Texto

Descripción generada automáticamente**

****

La línea roja nos indica donde se están descartando las líneas, y la línea azul nos indica el que se está descartando. En este caso, se están descartando una serie de metadatos como referencias, webs y puertos que no nos sirven. Como siempre son 10 referencias pues el número es estático.

1. ¿Qué se controla con el parámetro 'validation\_split'?, ¿por qué se ha elegido ese valor?, ¿qué ocurre si lo modificas? **(1 punto)**

Texto

Descripción generada automáticamente

El parámetro validation\_split está controlando la proporción de training y test del total de las muestras, es decir, que el conjunto de test es de tamaño 20% de las muestras.

La elección de este número puede variar según el conjunto de datos y los requisitos del problema. En muchos casos, se utiliza una división del 70-30 (entrenamiento-validación) o 80-20. En este caso, se ha elegido una división del 80-20.

Si se modifica el parámetro, cambiarían la proporción de los datos y, por consiguiente, el modelo se ajustaría más o menos a los datos y cambiaría las métricas de evaluación.

1. Imprime por pantalla un ejemplo (es decir, un elemento del array) de ‘train\_samples’, ‘val\_samples’, ‘train\_labels’ y ‘val\_labels’. A tenor de las etiquetas que se utilizan, ¿qué tarea crees que se está intentando entrenar? **(1 punto)**

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En este caso, se quiere entrenar un clasificador que me predecir, a través de un texto, la etiqueta a la que va asociada. Esta etiqueta tiene asociado semánticamente las carpetas de lo que tratan en orden alfabético, véase:

['talk.politics.misc', 'misc.forsale', 'rec.autos', 'sci.med', 'rec.sport.hockey', 'rec.sport.baseball', 'comp.graphics', 'rec.motorcycles', 'sci.crypt', 'talk.politics.mideast', 'alt.atheism', 'talk.politics.guns', 'comp.sys.mac.hardware', 'comp.os.ms-windows.misc', 'sci.space', 'sci.electronics', 'comp.sys.ibm.pc.hardware', 'soc.religion.christian', 'comp.windows.x', 'talk.religion.misc']

1. Con 'output\_sequence\_length' se establece un tamaño fijo para la salida de Vectorizer. ¿Por qué se necesita un tamaño fijo, y por qué se ha elegido el valor ‘200’? **(1 punto)**

Al trabajar con modelos de aprendizaje automático, es común que la entrada y salida tengan una dimensión fija. Esto se debe a que los modelos requieren entradas de tamaño constante para realizar operaciones eficientes y tener una representación consistente de los datos.

En el caso específico de 'output\_sequence\_length', se establece un tamaño fijo para la salida del Vectorizer para garantizar que todas las secuencias de texto tengan la misma longitud en su representación vectorial. Esto es necesario para alimentar adecuadamente los datos a un modelo, ya que generalmente se requiere que todas las entradas tengan la misma dimensión.

El valor específico de '200' para 'output\_sequence\_length' fue elegido probablemente en función de la longitud promedio de las secuencias de texto en el conjunto de datos o consideraciones específicas del problema. Es común seleccionar un valor lo suficientemente grande para abarcar la mayoría de las secuencias, pero no tan grande como para causar problemas de rendimiento o memoria. Es posible que se hayan realizado experimentos previos para determinar que un tamaño de 200 era adecuado para capturar la información relevante en el contexto del problema en cuestión.

Es importante tener en cuenta que el valor de 'output\_sequence\_length' puede variar según el conjunto de datos y el problema específico que se esté abordando. En algunos casos, puede ser necesario ajustar este valor y realizar pruebas para encontrar el tamaño óptimo que funcione mejor para el modelo y los datos específicos.

1. Indica cuál es la precisión del modelo en el conjunto de datos de entrenamiento y en el conjunto de datos de validación. ¿Qué interpretación puedes dar? Haz en este punto un análisis comparativo de los dos modelos ejecutados. **(1.5 puntos)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | TRAINING ACCURACY | TEST ACCURACY | |
| MODELO TRANSFORMERS | 0.9262 | Texto  Descripción generada automáticamente  0.7674 | |
| MODELO TRADICIONAL | 0.9658 | 0.7064 |

Aunque el modelo tradicional se ajuste mejor a los datos (produciéndose overfitting), se ajusta peor a los datos de test/validación, es decir, que su capacidad de generalizar es peor que el modelo Transformers. Esto nos lleva a que el modelo Transformers funcione mejor que el tradicional ya que se adapta mejor.

Esta técnica de training-test split nos ayuda a realizar un análisis de que modelo es mejor con respecto al otro, pero podríamos aplicar más técnicas como el k-folder splits para hacer más particiones como un 5-folder (validación cruzada) para seguir viendo más la capacidad de adaptación de ambas técnicas.

1. En la parte final del código se hace un análisis cualitativo de la salida. Explica el funcionamiento de este análisis e interpreta los resultados. Haz también en este punto un análisis comparativo de los dos modelos ejecutados. **(1 punto)**

|  |  |
| --- | --- |
| Texto  Descripción generada automáticamente | Texto  Descripción generada automáticamente |

En esta sección hemos implementado algunos ejemplos para comprobar la efectividad de ambos modelos prediciendo algunos nuevos, y se adaptan ambos a la clasificación requerida.

Se ha intentado encontrar algún caso de falso positivo, pero no se ha llegado a encontrar.

1. Explica algunas de las limitaciones que puedes encontrar al modelo entrenado. **(1.5 puntos)**
2. **Requisitos de recursos computacionales y memoria**: Los modelos Transformer se caracterizan por requerir una alta capacidad computacional y grandes cantidades de memoria debido a su arquitectura basada en atención y procesamiento paralelo de secuencias largas. Esto puede dificultar su implementación y entrenamiento en dispositivos con recursos limitados, lo que hace necesario el uso de supercomputadoras para desplegar eficientemente este tipo de modelos.
3. **Dependencia de datos etiquetados y recursos de entrenamiento**: Los modelos Transformer generalmente necesitan una gran cantidad de datos etiquetados para un entrenamiento efectivo. La obtención y etiquetado de conjuntos de datos de alta calidad puede resultar costoso y laborioso, especialmente en dominios especializados o con recursos limitados. Además, la falta de contexto semántico puede agravar este desafío.
4. **Limitaciones en la interpretación**: Aunque los modelos Transformer pueden lograr un alto rendimiento en diversas tareas, presentan limitaciones en cuanto a su capacidad para interpretar y explicar los resultados. Debido a la naturaleza de su arquitectura y al procesamiento a nivel de token, comprender cómo se llega a ciertas predicciones o qué características específicas se consideran relevantes puede resultar difícil.
5. **Tratamiento de secuencias de longitud variable**: Aunque los modelos Transformer pueden manejar secuencias de longitud variable, a menudo requieren técnicas adicionales como el uso de tokens especiales de inicio y fin, así como el relleno de secuencias más cortas para igualar la longitud. Estas técnicas pueden introducir ruido y dificultar el manejo de secuencias extremadamente largas o cortas. Por lo tanto, es necesario aplicar técnicas de procesamiento, como la tokenización o lematización, para obtener información relevante.
6. **Captura de relaciones a largo plazo**: A pesar de los avances logrados por los modelos Transformer en la captura de relaciones a largo plazo en secuencias, aún pueden enfrentar dificultades para capturar dependencias extremadamente prolongadas. En ocasiones, pueden perder información temporal o contextual en secuencias complejas, lo que puede resultar en un desequilibrio de clases y una precisión más baja para las clases menos representadas.
7. **Sensibilidad a datos de entrenamiento sesgados**: Al igual que otros modelos de aprendizaje automático, los modelos Transformer pueden verse afectados por sesgos presentes en los datos de entrenamiento. Si los datos de entrenamiento están sesgados hacia ciertas clases o contienen sesgos culturales o sociales, el modelo puede aprender y amplificar esos sesgos en sus predicciones.
8. ¿Qué sería necesario para que este modelo pueda interpretar textos en español? **(1 punto)**

Para entrenar un modelo Transformer en español, es necesario contar con un conjunto de datos de entrenamiento en español que capture las características y patrones específicos del lenguaje. Estos datos deben reflejar las diferencias lingüísticas entre el español y otros idiomas, como la concordancia gramatical y las reglas de género para los sustantivos.

Un enfoque común es utilizar un modelo de lenguaje pre-entrenado en español como punto de partida. Estos modelos han sido previamente entrenados en grandes cantidades de datos en español para aprender la estructura del lenguaje. Ejemplos de modelos de lenguaje pre-entrenados en español incluyen BERT en español, XLM-R en español y otros disponibles a través de la biblioteca de HuggingFace.

Además, se puede realizar un ajuste fino (fine-tuning) del modelo utilizando conjuntos de datos más pequeños y específicos en español, adaptándolo a tareas o dominios particulares como clasificación de texto o traducción. Para lograr un procesamiento del lenguaje natural más efectivo en español, también se requieren recursos lingüísticos específicos del español, como modelos de tokenización, etiquetado gramatical, diccionarios de palabras y recursos de lematización. Estos recursos ayudan al modelo a comprender y procesar el español de manera precisa.

Es importante llevar a cabo una evaluación continua y realizar ajustes en el modelo a medida que se utilice para interpretar textos en español. Esto implica corregir errores, ajustar los hiperparámetros del modelo y recopilar comentarios de los usuarios para mejorar su capacidad de interpretación en contextos específicos del español.

Entrega y evaluación

* Debe resolver la actividad en los Jupyter Notebook proporcionados (habrá uno por cada variante del ejercicio).
* Debe entregar un informe con la respuesta a las preguntas planteadas en el presente documento.
* Se valorará positivamente, la respuesta clara, breve y bien argumentada a las preguntas.
* Si se detecta **copia** entre alumnos en el código, en los comentarios o en las respuestas, todos los alumnos involucrados obtendrán una calificación para la actividad de cero puntos.

**Extensión máxima:** no hay restricciones en la extensión.

|  |  |  |  |
| --- | --- | --- | --- |
| Laboratorio: Word Embedings y Transformers para clasificación de texto | Descripción | Puntuación máxima  (puntos) | Peso  % |
| Pregunta 1 | La respuesta es válida y está bien argumentada. | 1 | 10% |
| Pregunta 2 | La respuesta es válida y está bien argumentada. | 1 | 10% |
| Pregunta 3 | La respuesta es válida y está bien argumentada. | 1 | 10% |
| Pregunta 4 | La respuesta es válida y está bien argumentada. | 1 | 10% |
| Pregunta 5 | La respuesta es válida y está bien argumentada. | 1 | 10% |
| Pregunta 6 | La respuesta es válida y está bien argumentada. | 1.5 | 15% |
| Pregunta 7 | La respuesta es válida y está bien argumentada. | 1 | 10% |
| Pregunta 8 | La respuesta es válida y está bien argumentada. | 1.5 | 15% |
| Pregunta 9 | La respuesta es válida y está bien argumentada. | 1 | 10% |
|  |  | **10** | **100 %** |