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)**

Al utilizar el comando para listar los archivos se entregan en un orden arbitrario. Por esa razón vamos a ordenar los archivos por orden alfabético. Sin importar que los nombres de los ficheros son cadenas numéricas, entonces:

#Algunos archivos de la categoria "com.graphics"

fnames = os.listdir(data\_dir / "comp.graphics")

fnames = sorted(fnames) # orden alfabetico

print("Number of files in comp.graphics:", len(fnames))

print("Some example filenames:", fnames[:5])

Ahora vamos a importar sPaCy y a crear el tokenizer:

# importar spacy tokenizer

import spacy

nlp = spacy.load("en\_core\_web\_sm")

tokenizer = nlp.tokenizer

Ahora vamos a calcular el numero de tokens de los primeros 15 archivos de comp graphics y su media:

# tokenizar los primeros 15 archivos de comp graphics y calcular su tamaño

fnames\_15 = fnames[:15] # seleccionar los primeros 15

tokens\_15 = [len(tokenizer(open(data\_dir / "comp.graphics" / fname).read())) for fname in fnames\_15] # calcular el numero de tokens de cada archivo

np.mean(tokens\_15) # media del numero de tokens

El resultado es: 297.2 tokens.

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)**

R/ El trozo de código que elimina las 10 primeras líneas de los ficheros se ha sombreado de amarillo. Debajo de esa línea de código se puso un print() para ver el contenido de la variable lines:

samples = []

labels = []

class\_names = []

class\_index = 0

for dirname in sorted(os.listdir(data\_dir)):

    class\_names.append(dirname)

    dirpath = data\_dir / dirname

    fnames = os.listdir(dirpath)

    print("Processing %s, %d files found" % (dirname, len(fnames)))

    for fname in fnames:

        fpath = dirpath / fname

        f = open(fpath, encoding="latin-1")

        content = f.read()

        lines = content.split("\n")

        lines = lines[10:] # acá se descartan las primeras 10 lineas

        print(lines)

        content = "\n".join(lines)

        samples.append(content)

        labels.append(class\_index)

    class\_index += 1

print("Classes:", class\_names)

print("Number of samples:", len(samples))

El resultado del print muestra lo siguiente:

Texto

Descripción generada automáticamente con confianza baja

Lo que nos permite entender que las primeras 10 lineas que se han eliminado representan todo el encabezado de los correos, en realidad, las líneas eliminadas varían de texto a texto debido a que no todas las mismas líneas se encuentran presentes en los correos, pero en general son las que contienen los siguientes campos:

Xref:

Path:

From:

Subject:

Message-ID:

Lines:

Sender:

Organization:

References:

Date:

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

La porción de código donde se establece el tamaño de los datos de validación se encuentra a continuación marcado de amarillo:

# Extract a training & validation split

validation\_split = 0.2

num\_validation\_samples = int(validation\_split \* len(samples))

train\_samples = samples[:-num\_validation\_samples]

val\_samples = samples[-num\_validation\_samples:]

train\_labels = labels[:-num\_validation\_samples]

val\_labels = labels[-num\_validation\_samples:]

El porcentaje de datos de validación es del 20% en este caso en particular. Por lo que los datos disponibles para entrenamiento son del 80%. El porcentaje de datos de validación se encuentra en el rango de valores aceptable, normalmente se ocupa un 60% de datos para entrenamiento, un 20% para validación del modelo y hacer cambios en él y un último 20% para evaluación del modelo.

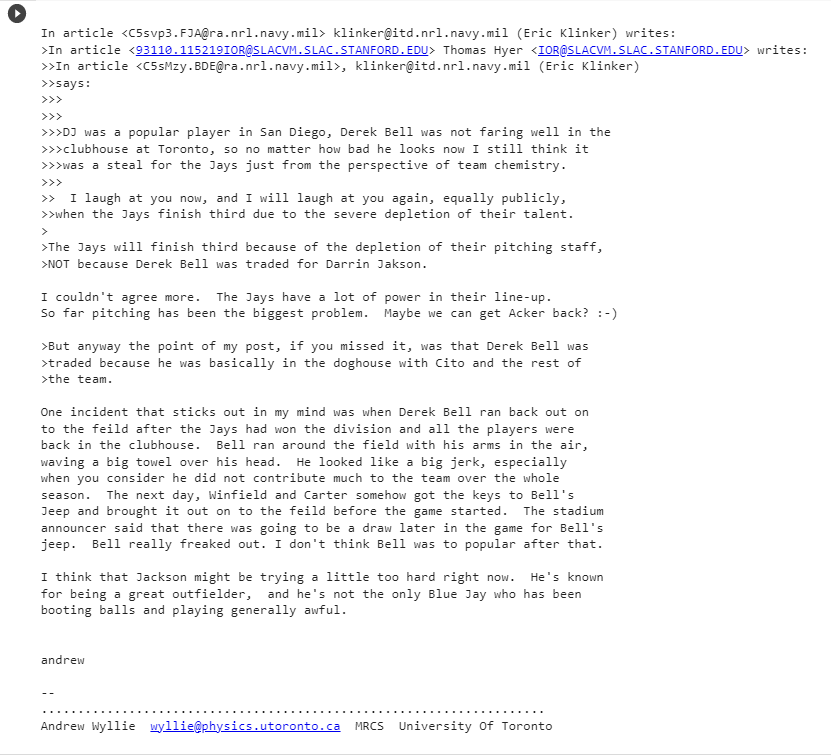
Si el porcentaje de datos de validación se aumenta, se corre el riesgo de tener muy pocas muestras para entrenamiento, por lo que la capacidad para predecir nuevos resultados del modelo puede ser seriamente afectada.

En cambio, si el porcentaje de datos de validación disminuye, se puede tener el riesgo de hacer overfitting al modelo, por lo que también podría afectar seriamente a la capacidad de predecir nuevos valores para datos no vistos.

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)**

# veamos un ejemplo de cada elemento de training y validacion

print(train\_samples[20]) # entrada



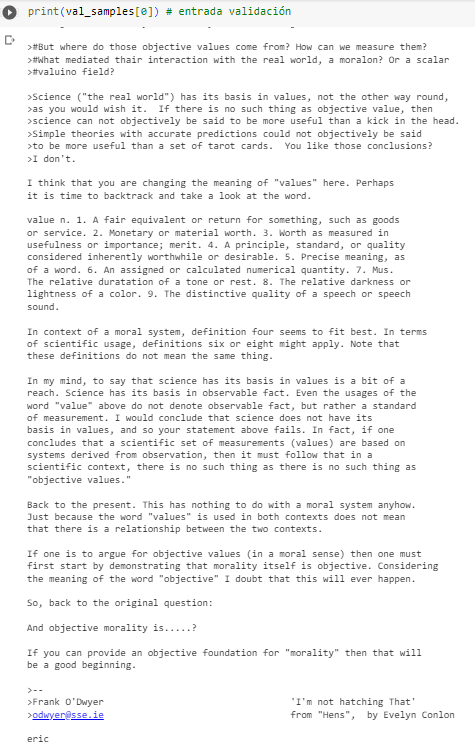
Y su label es:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

La cual coincide con el contexto del correo de training.

Veamos un ejemplo de validation:



Y la clase correspondiente es:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Por tanto, podemos concluir que la tarea que se pretende realizar en la clasificación de texto en una de varias categorías disponibles (en este caso, en total 20 categorías disponibles).

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)**

Lo que se busca con este parámetro es mantener el tamaño de las entradas de texto en una cantidad que sea manejable por el modelo, como vimos en la respuesta de la pregunta 1, el número de tokens en promedio anda entre 200-300 tokens aproximadamente, por lo que se busca mantener la salida de Vectorizer en una cantidad que permita procesar todas las entradas adecuadamente.

Las entradas con un valor mayor a 200 tokens serán truncadas y las menores de 200 tokens se les agregará padding para que todas tengan una vectorización del mismo tamaño.

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)**

Después de 20 epochs, obtenemos los siguientes resultados para ambos modelos:

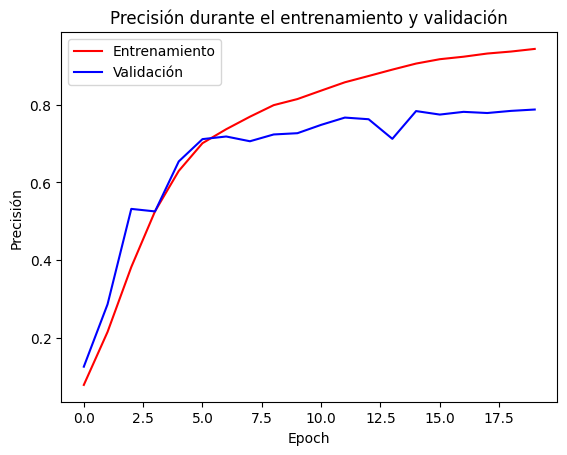
Para el modelo clásico:



Para el modelo de Transformers:



Gráfico, Gráfico de líneas

Descripción generada automáticamenteY haciendo gráfica de su accuracy y los para training y validation, obtenemos:

***Gráfica de accuracy para training y validation para los modelos: clásico (a la izquierda) y Transformers ( a la derecha).***

Gráfico, Gráfico de líneas

Descripción generada automáticamenteAhora obtenemos las gráficas de pérdida para ambos modelos:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

***Gráfica de loss para training y validation para los modelos: clásico (a la izquierda) y Transformers ( a la derecha).***

Luego de analizar ambos resultados podemos llegar a las siguientes conlusiones:

1. El accuracy en entrenamiento es mayor en el modelo clásico (0.9667 accuracy para clásico vs 0.9441 para transformers) sin embargo, en validation, el accuracy del modelo de transformers es mayor (0.7054 accuracy clásico vs 0.7879 transformers) por lo que la capacidad para generalizar en datos no vistos podría ser superior en el modelo de transformers. A pesar que ambos modelos aún son básicos se puede apreciar esta ventaja fundamental de los transformers.
2. Ambos modelos se estabilizan tanto en accuracy como en loss alrededor de los 10 epochs por lo que puede crearse un nuevo modelo reduciendo el numero de epochs para evitar el overfitting y variar otros hiperparámetros en cambio.
3. Respecto a las perdidas, estas van incrementando levemente alrededor de los 10 epochs por lo que es recomendable poner algunas callbacks para evitar el aumento de la perdida en ambos modelos.
4. 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)**

Comparemos las respuestas para clasificación de texto:

Para un texto sobre computer graphics y text modeling

Modelo clásico:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Modelo de transformers:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ambos modelos clasifican correctamente el texto presentado.

Veamos ahora el texto de religión.

Modelo clásico:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Vemos que el texto que habla sobre religión lo ha clasificado sobre un sistema operativo.

Modelo de Transformers:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Lo ha clasificado correctamente (un tema religioso misceláneo, dado que también hay otra categoría de religión cristiana).

Veamos un último ejemplo.

Modelo clásico:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Modelo de Transformers:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Vemos que ambos modelos no lo han clasificado correctamente, uno lo ha clasificado como política de Este Medio (Arabia, etc) y el modelo de Transformers lo ha clasificado como política, pero de armas. Es correcto afirmar que ambos modelos han clasificado adecuadamente que es un texto de política, pero dado que no se refiere a ninguna de las categorías dadas, lo correcto sería politics.misc.

La razón por la que pueden haber fallado en este caso es debido a que al hablar de política suelen existir palabras parecidas.

Sin embargo, se puede concluir que el modelo de transformer ha clasificado mejor en los tres casos. El modelo clásico se ha equivocado además en otro caso adicional.

1. Explica algunas de las limitaciones que puedes encontrar al modelo entrenado. **(1.5 puntos)**

De los modelos entrenados pueden verse algunas limitaciones:

1. La primera es que no existe un conjunto de datos de test sobre el cual hacer la evaluación final de la métrica considerada accuracy, por lo que se necesita partir los datos en training, test y validation.
2. El valor de accuracy para validation en ambos modelos, de approx. Con un 70%, ambas redes necesitan una modificación de sus arquitecturas agregando más capas o modificando hiperparámetros para producir mejores resultados. Por ejemplo, el modelo de red clásico podría mejorarse empleando más capas de embedding, densas y de dropout. El modelo de Transformers podría mejorarse probando con distintos valores de dimensión de embeddings embed\_dim y ff\_dim, además de la tasa de learning rate y otros hiperparámetros.
3. El valor de output\_sequence\_length=200 puede revisarse para evitar perder muchos tokens en la entrada y de esta manera vectorizar adecuadamente el texto.
4. ¿Qué sería necesario para que este modelo pueda interpretar textos en español? **(1 punto)**

Dado que estos modelos fueron entrenados con un corpus de texto en inglés, el dataset llamado "Newsgroup20" no podrán procesar texto en español adecuadamente. Se requiere un corpus en español para entrenar ambos modelos y puedan procesar y clasificar texto en 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 %** |