Universidad Rey Juan Carlos

TEMA 4 – HOJA DE EJERCICIOS I

Se plantean una serie de ejercicios para practicar con las redes neuronales vistas.

Ejercicio 1 – Red simple

Se propone un ejercicio para ver cómo entrenar un modelo simple de red neuronal para procesamiento de lenguaje natural y minería de texto utilizando la biblioteca Keras de Tensorflow. Se va a utilizar un conjunto muy básico de frases en español para clasificarlas según su sentimiento (positivo o negativo).

El conjunto de frases con su anotación es el siguiente:

- "Estoy un poco harto del día a día, nada mejora" -> Negativo
- "Hoy es un buen día" -> Positivo
- "No se te ve satisfecho con el trabajo" -> Negativo
- "Este paisaje es hermoso y bonito" -> Positivo

1) En este primer ejercicio no se va a realizar ningún tipo de preprocesamiento del texto. Las etiquetas (`labels`) serán 1 para la clase positiva y 0 para la clase negativa. Se define una longitud para los textos = 10. Se define un tamaño de vocabulario = 50.

Hay que ir siguiendo los pasos que se indican a continuación:

2) Construir el vocabulario (los tokens diferentes de los textos de entrada). Mostrar los tokens del vocabulario con su índice.

3) Construir la matriz de documentos-términos con el tamaño máximo elegido, donde el peso de cada término es el índice en el vocabulario (si no está se le pone valor 0).

4) Una vez preprocesado el texto, hay que definir el modelo. Se va a utilizar la clase `Sequential`, que permite añadir diferentes capas de forma secuencial.

5) La primera capa del modelo es la capa de embeddings, que se encargará de representar cada texto como una matriz de vectores. La matriz tendrá como número de filas el valor de la variable `max_length`, es decir, la longitud de las

secuencias de entrada. Como número de columnas, tendremos que decidir que dimensión queremos utilizar para representar los vectores de los tokens. En este ejemplo, vamos a utilizar un tamaño de vector pequeño (`vector_size` = 8), porque estamos con un ejemplo de juguete (las dimensiones que se suelen utilizar más son 100, 200 o 300).

La capa Embedding lo que hará será inicializar una matriz por cada texto. Como se ha dicho antes la matriz, tendrá una dimensión de `max_length x vector_size`.

En este ejercicio, la matriz se inicializa con pesos aleatorios, pero se podría inicializar la matriz a partir de un modelo pre-entrenado de word embeddings (lo veremos en otro ejercicio).

Hay que añadir una capa densa para la salida. En este caso, al ser una salida binaria como función de activación podemos utilizar sigmoid. Devolverá una probabilidad, que si es cercana a 1 entonces la capa de salida devolverá 1, 0 en otro caso.

6) Compilar el modelo, al hacer esto hay que indicar el optimizador de pesos a utilizar, la función de pérdida/error y las métricas que se van a utilizar en cada epoch.

   - Optimizador (optimizer): encargado de cambiar los atributos de la red neuronal como los pesos y la tasa de aprendizaje para reducir las pérdidas (e.g. 'adam', o 'rmsprop').
   - Función de pérdida (loss): mide cómo de bien el modelo está haciendo sus predicciones comparadas con los valores reales. El objetivo del entrenamiento es minimizar esta función.
   - Evaluación (metrics): métricas utilizadas para evaluar el rendimiento del modelo. No se utilizan para entrenar el modelo, pero son importantes para analizar cómo está funcionando el modelo.

7) Una vez que se ha configurado y compilado el modelo, se puede entrenar. Para ello, simplemente hay que llamar al método `fit`. Este método recibe los siguientes argumentos:

   - Los textos preprocesados y almacenados en la variable `encoded_train_pad`, y sus `labels` correspondientes, que el modelo utilizará para medir el error y aprender.
   - El número de epochs y el tamaño del batch. En este ejercicio, utilizaremos 5 epochs y un tamaño de batch de 32. Aunque lo razonable es probar con diferentes valores para ver cuáles proporcionan los mejores resultados.

- Los datos de validación. Si no se tiene conjunto de validación explícito, en el método `fit` (para entrenar el modelo) se puede especificar qué porcentaje del training se usaría como conjunto de validación. Para ello se usa el argumento `validation_split`.
- Además, también vamos se podría usar `EarlyStopping` que permite parar el entrenamiento si después de un número *n* de epochs (patience), el error sobre el conjunto de validación no ha disminuido.

8) Por último, se valida el modelo con un conjunto de frases de prueba:
   - "No fui al estreno de la película porque nadie me quería acompañar"
   - "Envidio de buena manera a los que tienen la oportunidad de ir mañana al estadio"
   - "Se nos está volviendo costumbre del domingo por la noche, ver el episodio anterior de SNL y eso me hace recibir el lunes con mejor humor"
   - "Al final decidí no ir al cine porque estaba cansada"

Nota: al tener una inicialización de pesos aleatoria, si se ejecuta varias veces sin cambiar la configuración, los resultados pueden ser distintos.

Ejercicio 2 – RNN

Se propone entrenar modelos simples de redes neuronales recurrentes (RNN por sus siglas en inglés) sobre un conjunto de noticias clasificadas como verdaderas o falsas (fake news).

Para ello, se utilizará el ficherto **train.xlsx** del conjunto de datos The *Spanish Fake News Corpus*. Puedes obtener dicho fichero a través del siguiente enlace:

https://github.com/jpposadas/FakeNewsCorpusSpanish/tree/master

1) **Carga del dataset -** Descarga el anterior dataset y obtén los títulos de las noticias (columna *Headline*), el contenido de estas (columna *Text*) y sus etiquetas (columna *Category*). Clasifica las etiquetas en 0 si la noticia es real (*True*) y 1 si es falsa (*Fake*).

2) **Análisis exploratorio -** Analiza la distribución del número de palabras tanto en los títulos de las noticias (frases cortas) como en su contenido (varias frases). Este análisis será utilizado posteriormente para decidir el tamaño máximo a introducir por cada documento.

3) **Separación del dataset -** Divide los títulos de las noticias y su contenido en sus correspondientes conjuntos de entrenamiento, validación y test. Recuerda que los conjuntos de entrenamiento serán usados posteriormente tanto para la construcción de los vocabularios como para el entrenamiento de las redes neuronales.

4) **Construcción del vocabulario -** Construye los vocabularios tanto para los títulos de las noticias como para su contenido. Realiza un preprocesamiento básico de las palabras que los componen.

5) **Codificación de los textos** - Construye las matrices de documentos-términos, donde cada palabra es representada por su índice en el vocabulario correspondiente. Para la elección del tamaño máximo, usa el análisis anteriormente realizado sobre la distribución del número de palabras.

6) **Entrenamiento sobre los títulos**
   a. Empleando la capa SimpleRNN() de Keras, diseña y entrena una red neuronal recurrente para los títulos de las noticias. Entrena la capa de embeddings desde cero. Usa los correspondientes conjuntos de validación anteriormente creados para evaluar cómo se van entrenando los modelos diseñados.
   b. Prueba distintas configuraciones de la red neuronal (número de neuronas, número de dimensiones de la capa de Embeddings, Dropout / Recurrent Dropout…).
   c. Con los modelos entrenados sobre los títulos de las noticias, realiza predicciones sobre su correspondiente conjunto de test y analiza los resultados obtenidos.

7) **Entrenamiento sobre los contenidos**
   a. Repite el proceso anterior de entrenamiento de redes neuronales recurrentes para el contenido de las noticias usando la capa SimpleRNN() de Keras.
   b. Nuevamente, experimenta sobre distintas configuraciones y diseños de entrenamiento.
   c. Con los modelos de redes neuronales entrenados sobre el contenido de las noticias, realiza predicciones sobre su correspondiente conjunto de test. Analiza los resultados obtenidos y compáralos sobre los anteriormente obtenidos con los títulos.

8) **Entrenamiento con embeddings de Word2Vec**
   a. En vez de entrenar los embeddings desde cero, emplea un modelo de Word2Vec en español para su inicialización. Puedes usar el modelo “SBW-vectors-300-

min5.txt" (https://www.kaggle.com/datasets/rtatman/pretrained-word-vectors-for-spanish).

b. Inicializa los embeddings de las redes neuronales diseñadas anteriormente usando Word2Vec. Prueba distintas configuraciones de redes neuronales y analiza los resultados obtenidos.

Ejercicio 3 - LSTM

En este ejercicio se propone entrenar redes neuronales recurrentes con memoria a largo y corto plazo (LSTMs por sus siglas en inglés) sobre el mismo conjunto de datos.

- Repite los pasos realizados en el Ejercicio 1 tanto para los títulos de las noticias como para su contenido completo, empleando en esta ocasión la capa **LSTM()** de la librería Keras.
- Experimenta nuevamente distintas configuraciones de la red neuronal, modificando sus hiperparámetros (número de unidades LSTM, dimensiones de la capa de Embeddings, regularización con Dropout o Recurrent Dropout, etc). Prueba a inicializar los embeddings de forma aleatoria (entrenamiento desde cero) o por medio del uso del modelo de Word2Vec "SBW-vectors-300-min5.txt".
- Analiza los resultados, comparando las diferentes configuraciones de LSTM experimentadas y los resultados obtenidos durante el Ejercicio 1.