Trabajo práctico: Redes Recurrentes y Representaciones Incrustadas

Ph. D. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Ingeniería en Computación,
PAttern Recognition and MAchine Learning Group (PARMA-Group)

28 de octubre de 2022

El presente proyecto introduce el uso de redes recurrentes para la clasificación de mensajes en correo deseado o correo no deseado.

- **Fecha de entrega: 12 de Noviembre**
- **Modo de trabajo**: Grupo de tres/dos personas.
- **Tipo de entrega:** digital, por medio de la plataforma TEC-digital.

Para el presente trabajo práctico usted utilizará el conjunto de datos *SMSSpamCollection* para resolver la clasificación binaria en dos clases de mensajes de texto entre mensajes no deseados (*spam*) y mensajes rutinarios (*ham*). *SMSSpamCollection dataset*, disponible en `https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection` el cual contiene 5571 mensajes de texto, con sus etiquetas de *spam* o *ham*. A continuación, se muestran observaciones de ejemplo para ambas clases:

```
ham Siva is in hostel aha:-. ham Cos i was out shopping wif darren jus
now n i called him 2 ask wat present he wan lor.
Then he started guessing who i was wif n he finally guessed darren lor.
spam FreeMsg: Txt: CALL to No: 86888 & claim your reward of 3 hours talk
time to use from your phone now!
subscribe 6GBP/ mnth inc 3hrs 16 stop?
spam Sunshine Quiz!
Win a super Sony DVD recorder if you can name the capital of Australia?
Text MQUIZ to 82277.
```

1. (100 puntos) Red LSTM

1. Implemente la siguiente arquitectura de una red LSTM:

a) **(40 puntos)** Como primer etapa de preprocesamiento se implementarán dos funciones *preprocesar_documento_1* y *preprocesar_documento_2*. En la primer versión del preprocesamiento de los documentos (*preprocesar_documento_1*), usted implementará la eliminación de mayúsculas y la eliminación de *stop-words* utilizando la librería *nltk*. En la segunda versión del preprocesamiento, además de tales funcionalidades usted implementará la lematización de las palabras, también usando NLTK.

   1) Muestre un ejemplo con una entrada escogida del pre-procesamiento con ambos enfoques, y explique brevemente los posibles efectos de utilizar el segundo enfoque al primero.

b) **(20 puntos)** Utilice como base el código provisto. La capa embebida en este caso es la definida dentro de la librería de pytorch. Implemente $D = 20$, $D = 100$ dimensiones del vector incrustado. **Utilice el segundo enfoque de preprocesamiento**. Muestre los resultados en una tabla con sus medias y desviaciones estándar y coméntelos. Compare los resultados respecto a lo obtenido anteriormente con la arquitectura.

c) **(20 puntos)** Ejecute el experimento anterior usando el **primer enfoque de preprocesamiento**, usando $D = 20$, $D = 100$ dimensiones del vector incrustado. Muestre los resultados en una tabla con sus medias y desviaciones estándar y coméntelos.

2. (30 puntos extra) Perceptrón multi-capa

En este modelo, usted utilizará el conjunto de datos mencionado para construir un perceptrón de 3 capas. A continuación se detallan las especificaciones del modelo a construir.

1. **(10 puntos)** Para la extracción de características se implementará un extractor *aprendido* a partir del conjunto de datos de entrenamiento con la arquitectura *word2vec*. Para ello utilizará la librería *gensim*. Entrene al extractor de características usando el corpus de entrenamiento completo. Para ello defina una función *extract_features_dataset(model, preprocessed_dataset, max_length_words = 100, num_features = 20)* la cual retorne un tensor *features_dataset* con $N \times D$ dimensiones, donde N es la cantidad de observaciones de la muestra, y D la cantidad de dimensiones seleccionadas para el vector incrustado (*num_features*).

   a) Muestre un pantallazo con un ejemplo de la corrida, y comente los problemas encontrados para implementar tal conversión de los datos a vectores incrustados.

2. **(20 puntos)** Implemente una red neuronal usando el paquete *torch.nn* con la cantidad de neuronas en la capa de entrada necesarias, y una cantidad M de neuronas en la capa oculta a seleccionar.

a) Escoja una función de activación a la salida, y una función de pérdida adecuada y justifique su decisión.

b) Implemente las funciones *train_model* y *test_model* según sea necesario utilizando como base el código provisto anteriormente. Comente los cambios.

   1) Para la función *test_model* puede utilizar el paquete *sklearn.metrics*.

c) Realice las pruebas de tal red neuronal usando $M = 0{,}7\ D$ con $D = 20$, $D = 100$ dimensiones del vector incrustado. Utilice el segundo enfoque de preprocesamiento en tales pruebas. Realice 10 corridas por cada tratamiento utilizado y mida la tasa de aciertos, y el F1-score. Muestre los resultados en una tabla con sus medias y desviaciones estándar y coméntelos.