PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
IIC-3697 APRENDIZAJE PROFUNDO

TAREA 2: ANÁLISIS DE TEXTO CON CNNS Y RNNS

Fecha de Entrega: Lunes 4 de Junio

Estimad@s, en esta actividad tendrán la oportunidad de poner en práctica sus conocimientos sobre aprendizaje profundo (deep learning). En particular, podrán experimentar con modelos para el análisis de texto. Para el desarrollo de la tarea nuevamente pueden utilizar la plataforma Google Colaboratory. En términos de la plataforma de software podrán usar Keras, o innovar usando TensorFlow o Pytorch (en clases veremos detalles de estas 2 herramientas).

Parte 1: Calentando Motores con Redes Recurrentes (20%).

Como comentamos en clases, las redes recurrentes nos permiten modelar secuencias, es decir, permiten capturar explícitamente relaciones temporales o espaciales en los datos. En esta actividad evaluaremos esta capacidad mediante el uso de un ejemplo incluido en las librerías de Keras. Específicamente, usaremos una RNN para predecir el flujo mensual de pasajeros en vuelos internacionales en EE.UU. Para esto usaremos el set de datos "international-airline-passengers.cvs" disponible en el sitio web del curso. Este set de datos contiene la demanda mensual de pasajes internacionales para un período de 12 años, correspondiente a Enero 1949 a Diciembre 1960. La figura muestra una gráfica de este set de datos.

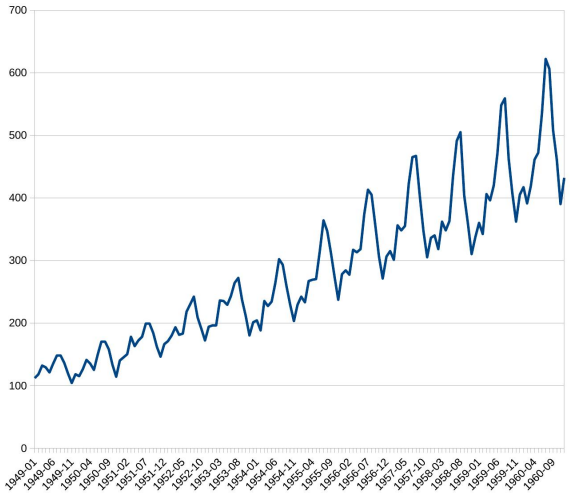

Figura 1: Total mensual de pasajeros en vuelos internacionales en EE.UU para el período Enero 1949 a Diciembre 1960. Cifras en miles de pasajeros.

La tendencia creciente de la curva muestra el aumento de pasajeros a lo largo de los años. Adicionalmente, cada año presenta peaks característicos causados por aumentos puntuales de la demanda debido al período de vacaciones y festividades especiales.

Para este set de datos, usaremos el período Enero 1949 a Diciembre 1956 para aprender un modelo del tipo RNN, y el resto de los datos para probar la capacidad predictiva del modelo obtenido. Específicamente,

definiremos un modelo base, en el cual consideraremos el problema de predecir la demanda en el mes t según la información de demanda en los 4 meses anteriores. Luego estudiaremos variantes de este modelo base.

El siguiente código muestra la definición en Keras de una red neuronal recurrente para el modelo base.

```
modelRNN = Sequential()
modelRNN.add(SimpleRNN(5,input_dim=1,input_length=4,return_sequences=False))
```

Primeramente se define el container, que en este caso es denominado *modelRNN*. Luego se define la red recurrente según los siguientes parámetros:

- El primer parámetro indica la dimensionalidad del espacio de características del estado intermedio o oculto, en este caso dimensionalidad 5.
- *input_dim* indica la dimensionalidad de la entrada, en este caso 1 (demanda mensual).
- *input_length* indica el largo de la secuencia de entrada, en esta caso 4 meses anteriores.
- *return_sequences=False* indica que sólo el último nodo de la secuencia intermedia genera una salida (ver figura 3).

La siguiente figura muestra un diagrama de la RNN definida por el código anterior:

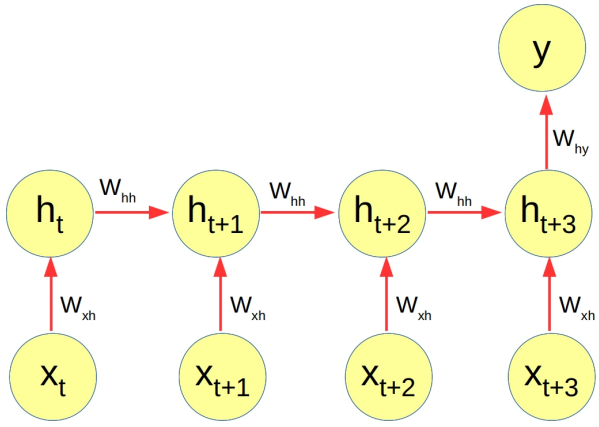

Figura 2: Arquitectura de la red recurrente definida para la red recurrente base.

Al analizar los embeddings podemos calcular el número de parámetros que es necesario ajustar durante el aprendizaje. Por ejemplo, la entrada corresponde a la medición de demanda de cada mes, por tanto, tiene dimensión 1 (*input_dim=1*). Por su parte, el estado intermedio h_t tiene dimensionalidad 5. Así el embedding W_{xh} transforma de dimensionalidad 1 a 5, por tanto, W_{xh} consiste de una matrix de 5x1 valores, es decir, $W_{xh} \in \mathbb{R}^{5x1}$, 5 parámetros.

Actividad 1

Indique el total de parámetros del modelo anterior. Puede verificar su cálculo utilizando la función *summary()*. Fundamente su respuesta.

Se decide modificar la dimensionalidad del estado intermedio a un valor 4, ¿Cómo se afecta el número de parámetros?. Pruebe y fundamente su respuesta.

Actividad 2

Para realizar la predicción de la demanda podemos agregar a la red anterior una capa densa. Dado que queremos predecir el valor de una variable que toma valores en el eje de los número reales, debemos utilizar sólo 1 neurona como salida y no incorporar función de activación. De esta manera el modelo base de predicción queda dado por:

```
# create and fit the RNN
modelRNN = Sequential()
modelRNN.add(SimpleRNN(5,input_dim=1,input_length=4,return_sequences=False))
modelRNN.add(Dense(1))
```

El archivo AirlinePrediction.py, disponible en el sitio web del curso, contiene el código anterior y las funciones necesarias para cargar el archivo de datos y mostrar los resultados obtenidos. Use este código para probar el rendimiento del modelo base. En términos de la presentación de resultados, la salida del archivo AirlinePrediction.py indica el error medio cuadrático en el set de entrenamiento y test. Además muestra una gráfica que incluye los datos originales (curva azul), la predicción en datos de entrenamiento (curva verde) y la predicción en datos de test (curva roja).

Ejecute el modelo base varias veces, ¿Por qué no se obtiene siempre el mismo resultado?. Fundamente su respuesta.

Luego pruebe las siguientes variantes al modelo base y analice su resultados:

- Modifique el largo de la secuencia de entrada. Indique como este cambio afecta la precisión de la predicción.
- Agregue una nueva capa densa de 100 unidades. Indique como este cambio afecta la precisión de la predicción.
- Agregue una capa de Dropout. Indique como este cambio afecta la precisión de la predicción.

Actividad 3

La red anterior implementa una red convolucional tradicional. Como comentamos en clases, este tipo de modelo presenta limitaciones para ajustar los parámetros de la red utilizando métodos de descenso de gradiente (problema de desvanecimiento de gradiente o vanishing gradient problem). Redes recurrentes tipo LSTM o GRU ofrecen soluciones más estables. Keras ofrece implementaciones de ambas alternativas, siendo muy simple su uso, basta cambiar el nombre en el llamado a la función.

Compare el rendimiento de un modelos LSTM con respecto al modelo tradicional (SimpleRNN). Considere el rendimiento en la predicción y el número de parámetros utilizado.

Parte 2: Modelación de Documentos (40%).

Como comentamos en clases, RNNs son una excelente herramienta para capturar relaciones entre los componentes de una secuencia. La secuencia a modelar puede seguir distintos tipos de ordenamiento, por ejemplo, temporal, espacial, ranking, u otro. Por tanto, estos modelos pueden ser utilizados en una gran variedad de aplicaciones.

Como principio fundamental, una RNN es capaz de capturar relaciones relevantes entre los datos de entrada. Por ende, aparte de tareas de clasificación o predicción, el modelo resultante también puede ser utilizado para generar nuevas secuencias que respondan a los patrones del set de entrenamiento. Utilizando esta capacidad han surgido aplicaciones capaces de generar escritura manuscrita [1], leyendas explicativas de imágenes [3], composiciones musicales [1] y pinturas con distintos sellos artísticos [2]. A modo de ejemplo, la siguiente figura muestra como una RNN transforma un simple bosquejo en una pintura postimpresionista.

En esta parte del laboratorio experimentarán con una RNN capaz de sintetizar textos que siguen el estilo de escritura de un conjunto de documentos determinado. En el sitio web del curso se publicarán algunas opciones, por ejemplo, todas las oraciones del "Quijote de la Mancha" o los textos de Wikipedia. Adicionalmente, puede utilizar los documentos que usted desee, lo importante es que el conjunto de entrenamiento cumpla 2 reglas fundamentales para el buen funcionamiento de una técnica de aprendizaje profundo: i) Una

[1] http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/
[2] https://github.com/alexjc/neural-doodle

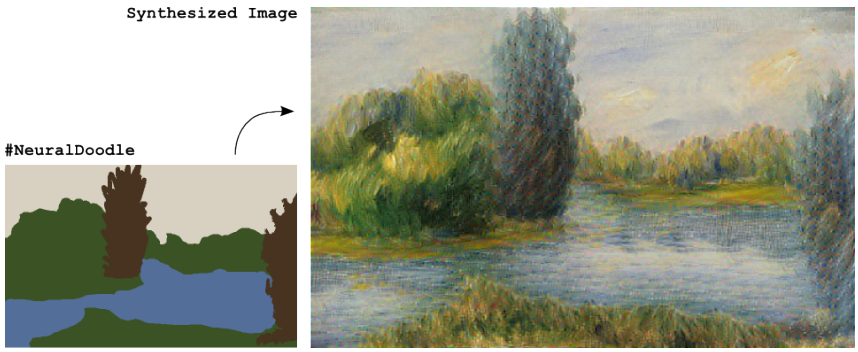

Figura 3: Ejemplo de una RNN que transforma un simple bosquejo en una pintura con cierto estilo artístico.

gran cantidad de ejemplos de entrenamiento (cientos de miles o millones de registros), ii) Diversidad y representatividad, los ejemplos de entrenamiento deben ser diversos y representativos del dominio elegido.

Si bien pueden existir variaciones, una posible estrategia de generación es la que vimos en clases. Los textos de entrenamiento son divididos en secuencias de caracteres. Cada una de estas secuencias de largo n+m representa un ejemplo de entrenamiento, donde el objetivo es usar los primeros n caracteres para predecir los siguientes m. Por ejemplo, las secuencias utilizadas pueden ser de largo 100 y ser tomadas con un paso (stride) de 1 caracter desde el texto de entrada. De esta manera el objetivo será predecir el caracter 100 utilizando los 99 anteriores. Para efectos de la RNN cada caracter puede ser codificado inicialmente con una representación one-hot.

Actividad 4

Utilizando sus conocimientos de aprendizaje profundo diseñe un modelo que permita modelar los documentos del dominio seleccionado. Este modelo debe ser esquematizado según un diagrama de bloques que debe incluir en su informe de tarea.

Actividad 5

Utilizando Keras, TensorFlow o Pytorch implemente el modelo propuesto. Luego entrene este modelo usando los documentos seleccionados.

Actividad 6

Genere nuevos documentos con su modelo. Incluya en su informe algunos ejemplos que encuentre ilustrativos de la operación de su modelo. Comente sus observaciones.

Parte 3: Preguntas y Respuestas en Documentos (40%).

En esta pregunta exploraremos las potencialidades de técnicas de aprendizaje profundo para aplicaciones de pregunta y respuesta en texto (question and answers, Q&A using text). En particular, utilizaremos el set de datos SQuAD publicado por investigadores de la Universidad de Stanford (Stanford Question Answering Dataset, `https://rajpurkar.github.io/SQuAD-explorer/`). Este set consiste de más de 100.000 pares pregunta-respuesta construidos en forma manual en base a documentos de Wikipedia. Como característica relevante, para este set la respuesta a cada pregunta consiste de un segmento del texto correspondiente. En el siguiente link pueden explorar este set de datos y ver algunos ejemplos: `https://rajpurkar.github.io/SQuAD-explorer/explore/1.1/dev/`.

Actividad 7

Utilizando sus conocimientos de aprendizaje profundo y el set de entrenamiento de SQuAD, diseñe un modelo que permita entregar respuestas a las consultas del set de desarrollo (DevSet). Este modelo debe ser esquematizado según un diagrama de bloques que debe incluir en su informe de tarea. Como modelación inicial de las palabras puede utilizar codificaciones tales como: i) One-hot (vista en clases), ii)

Word2Vec [4], iii) GloVe [2], o equivalentes. Para esto puede usar las librerías del toolkit Gensim (`https://radimrehurek.com/gensim/`).

Actividad 8

Utilizando Keras, TensorFlow o Pytorch implemente el modelo propuesto. Luego entrene este modelo usando el set de entrenamiento de SQuAD disponible en el sitio web del curso.

Actividad 9

Pruebe el rendimiento de su implementación utilizando el set de desarrollo de SQuAD (devset) disponible en el sitio web del curso. Para evaluar el rendimiento de su algoritmo utilice las métricas: i) Exact match y ii) F1-score, sugeridas en [5]. En el sitio web del curso podrá encontrar el código evaluate-v1.1.py que puede utilizar como referencia para realizar la evaluación. Documente y analice brevemente sus resultados.

Bibliografía

[1] A. Graves. Generating sequences with recurrent neural networks. In *arxiv.org/pdf/1308.0850v5.pdf*, 2013.

[2] C. Manning J. Pennington, R. Socher. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[3] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *In Proceedings of Workshop at ICLR*, 2013.

[5] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. https://arxiv.org/abs/1606.05250, 2016.