### **📚 Resumen Clase 6 – Arquitectura Encoder–Decoder y Secuencia a Secuencia**

#### **🧠 Contenido Principal**

**1. Modelos Secuencia a Secuencia (Seq2Seq)**

* Generalización de modelos many-to-many, donde **entrada y salida pueden tener distinta longitud**.
* Usado para **traducción, chatbots, resumen automático**, etc.
* Compuesto por dos partes:  
  + **Encoder**: procesa toda la secuencia de entrada y produce un estado oculto (latente).
  + **Decoder**: genera la secuencia de salida en forma autoregresiva (un token a la vez), usando el estado latente y lo generado anteriormente.

**2. Detalles técnicos del modelo**

* Se explicó cómo usar **LSTM** (o GRU) en encoder y decoder.
* El decoder necesita:  
  + Un token especial de **inicio de secuencia** (Start of Sequence – SOS).
  + Un token de **fin de secuencia** (End of Sequence – EOS) para saber cuándo parar.
* Durante **entrenamiento**, el decoder recibe la secuencia de salida real (desplazada).
* Durante **inferencia**, el decoder **genera token por token** hasta producir el EOS.

**3. Implementación en Keras**

* Se construyó el modelo en tres partes:  
  + Modelo completo para **entrenamiento**.
  + Submodelo del **encoder** para inferencia.
  + Submodelo del **decoder** para inferencia (requiere reinyectar estados de memoria H y C).
* Se explicó la importancia de usar correctamente los initial\_state y de cómo armar una función de inferencia paso a paso.

**4. Traducción de texto (caso práctico)**

* Se armó un ejemplo de traducción **inglés → español** usando pares de oraciones.
* Se cargó un corpus, se tokenizó por separado para encoder y decoder.
* Se usaron **capas de embedding** para representar texto de entrada y salida.
* Se entrenó una red seq2seq básica y se implementó una **función de inferencia** para traducir nuevas frases.

**5. Observaciones del entrenamiento**

* Se mostró cómo entrenar el modelo con model.fit().
* Se discutió un fenómeno de **divergencia entre entrenamiento y validación** por tener secuencias en validación que no estaban en el entrenamiento.
* Se destacó la importancia del **tokenizador**, que debe ir siempre asociado al modelo.

#### **🧪 Ejemplo final**

* Se generaron traducciones con el modelo entrenado.
* Aunque no perfectas, se observó que el modelo **aprende a detenerse** y a generar salidas gramaticales coherentes.
* Se discutió cómo mejorar resultados aumentando datos, embeddings preentrenados, etc.

### **🧩 Desafío 4 (planteado en esta clase)**

El **Desafío 4** es construir un modelo de **traducción automática** como el del ejemplo visto en clase:

* Arquitectura encoder-decoder entrenada con pares de frases.
* Incluir tokenización, embeddings, entrenamiento y función de inferencia.
* Presentar resultados, incluyendo ejemplos de traducciones generadas.