Actividad 4: NLP moderno con Deep Learning

Teoría

1. ¿Qué problema tienen las redes neuronales clásicas (MLP) para procesar secuencias?
   1. Las MLP (redes neuronales multicapa) no están diseñadas para recordar información de entradas anteriores. Esto significa que, al procesar secuencias, no pueden retener el contexto de elementos anteriores, lo que es crucial en tareas como traducción de idiomas o análisis de sentimientos donde el significado puede depender de palabras anteriores. Requieren que todas las entradas tengan la misma forma. Esto obliga a usar técnicas como el padding (relleno), que pueden introducir información irrelevante y dificultar el aprendizaje. Durante el entrenamiento, las MLPs pueden sufrir problemas de desvanecimiento del gradiente, donde los gradientes se vuelven demasiado pequeños para realizar ajustes significativos en los pesos. Tienen dificultades para capturar relaciones a largo plazo en los datos. Los datos de entrada se tratan de manera independiente, lo que significa que no se aprovecha la estructura de las secuencias.
2. Explique que es una neurona recurrente y cuál es la diferencia con una neurona clásica.
   1. Una RNN puede procesar secuencias de tamaños arbitrarios en vez de tamaño fijo. Además de estado de entrada y salida, tiene un estado oculto, donde la salida del estado anterior es la entrada del siguiente.
3. Explique como una red neuronal recurrente procesa una secuencia de texto a través del tiempo.
   1. Output de la neurona a cada time step t es una función que tiene como input todos los time step previos. La red conserva cierto estado a través de los time steps, a esto se lo llama memory cell. Solo puede aprender aprox. 10 steps dependiendo de la tarea. El estado de la celda a cada time step se lo describe como h(t) (Hidden)
4. Enumere las arquitecturas de redes recurrentes y defina cada una.
   1. Sequence to vector
      1. Input secuencia de inputs
      2. Last Output. Útil para Text classification: Hate(0) y Love(1)
   2. Vector to Sequence
      1. Input un vector. Una imagen resultada de una CNN.
      2. Output secuencia: Subtitulado de esa imagen.
   3. Sequence to sequence
      1. Input secuencia
      2. Output secuencia. Ejemplo traducciones.
5. ¿Qué problemas tiene una red recurrente simple? ¿Qué alternativas surgieron para superarlos?
   1. Gradientes inestables. Se puede aliviar con técnicas:
      1. Recurrent dropout
      2. Recurrent layer normalization
   2. Memoria muy limitada. Se puede extender con dos variantes
      1. LSTM (Long Short Term Memory): Entrenamiento converge mucho más rápido. Detecta patrones a largo plazo en los datos. Requiere mucho más dato que una RNN simple.
      2. GRU (Gated Recurrent Units): Introduce la red encoder y decoder. Se descarta un estado para poder guardar otro. Memoria más limitada que LSTM, necesita gran cantidad de tiempo para aprender patrones en secuencias de 100 frames o más.

Práctica

En esta actividad práctica aplicar los conceptos vistos en la teoría. Para ello realizar una copia, ejecutar y analizar el colab: 12 - RNN, LSTM y GRU.ipynb para responder las siguientes preguntas a modo de reflexión:

1. ¿Qué problema busca solucionar en el colab? Definir alcance y meta.
   1. El objetivo es tomar múltiples reviews de películas y clasificar la emoción de la review como positiva o negativa.
2. ¿Qué función cumple la capa TextVectorization y Embedding?
   1. TextVectorization de Keras: realiza tokenización, crea vocabulario, vectoriza, maneja palabras desconocidas y normaliza.
   2. Embedding de Keras: se utiliza para convertir enteros (que representan palabras o tokens) en vectores de representación densa, comúnmente conocidos como embeddings. Se utiliza generalmente después de una capa de vectorización para convertir las palabras en índices enteros. Toma como entrada una secuencia de enteros y devuelve una representación densa correspondiente a esos enteros.
3. ¿Cómo variamos las diferentes arquitecturas de una capa recurrente?
   1. En el codigo que se setea una capa GRU con 128 units (Tamaño del embeddings) para poder recibir el vector de features. Por default todas las RNN utilizan tanh como función de activación y devuelven solo el ultimo output de la secuencia. Si queremos devolver toda la secuencia debemos indicarle el hiperparametro return\_sequences en true
4. ¿Qué técnicas podemos utilizar para tratar palabras que no están en nuestro vocabulario con este tipo de redes recurrentes?
   1. Masking: Cuando hay un valor perdido en nuestra secuencia (palabra no presente en el encoding) se puede enmascarar. Por lo tanto, podemos enmascararlas manualmente o simplemente en la embedding layer podemos pasarle el hiperparametro mask\_zero en True.
   2. Ragged tensors: usar una lista de arrays de diferente dimension, quitando las features que esten en 0.

NLP moderno con Transformers

1. ¿Qué es el framework Encoder y Decoder con redes recurrentes y para qué sirve?
   1. Cualquier tipo de red puede ser encoder y decoder. En este caso usamos una celda RNN. Una palabra ingresa por vez y se genera una palabra como output.
2. ¿Qué problemas tiene este framework?
   1. Esta arquitectura genera un cuello de botella en el last state que se pasa de un modelo a otro. El decoder necesita acceder a todos los estados anteriores para representar efectivamente el significado de la oración.
3. ¿Qué es un large language model?
   1. Es un modelo estadístico. Proporciona la distribución de probabilidad sobre una secuencia de palabras.
4. ¿Qué desarrollos impulsaron la aparición de los large language models?
   1. Los pipelines de Google llamados Transformers.
   2. Una técnica de transfer learning llamada ULMFiT.
5. ¿En qué consiste la arquitectura Transformers?
   1. Son modelos de secuencia a secuencia
   2. Como input recibe una secuencia de tokens
   3. Como output mapea otra secuencia de tokens
   4. Hecha para mejorar rendimiento y calidad de traducciones
6. ¿Qué es el mecanismo de atención y para qué sirve?
   1. El modelo procesa cada palabra del input mirando las otras posiciones de la secuencia para mejorar el encoding de la palabra que está en procesamiento. Permite al modelo dar más peso a las palabras cuando hay ambigüedad en las frases.
7. ¿Qué es un modelo auto regresivo y auto encoding?
   1. Auto regressive models
      1. Predecir la siguiente palabra en una oración (autocompletar) en base a contexto anterior.
      2. Natural Language Generation (NLG)
      3. Familia GPT
   2. Auto encoding models
      1. Comprensión y codificación integrales de secuencias completas de tokens.
      2. Natural Language Understanding (NLU)
      3. Familia BERT
8. Defina que es el modelo BERT. Indique qué tipo de modelo es y qué tareas cumple.
   1. Bidirectional Encoder Representation from Transformers.
   2. Es un encoder-based Transformer model.
   3. Toma input sequence (colección de tokens).
   4. Produce un encoded output sequence (Hidden representation).
   5. Tareas que resuelve
      1. Sequence classification
      2. Token classification
      3. Question Answering
      4. Multiple choice
9. Defina que es el modelo GPT. Indique qué tipo de modelo es y qué tareas cumple.
   1. Generative Pre-trained Transformer
   2. Auto-regressive Language Model
   3. Los decoders se entrenan con enormes corpus de datos.
   4. GPT solo toma el decoder de la arquitectura Transformer para su estructura.