CS-4053 Recommender System

Fall 2023

Lecture 13: Transformers in Recommender System

Course Instructor: Syed Zain Ul Hassan

National University of Computer and Emerging Sciences, Karachi

Email: zain.hassan@nu.edu.pk

Transformer: Background

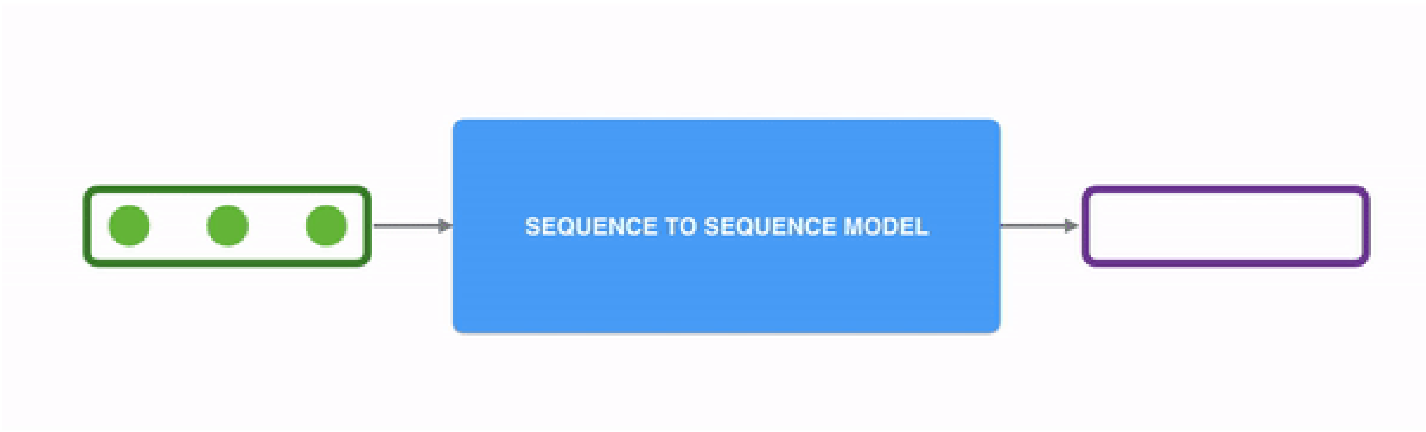
- In some problems, we need our neural network to "*remember*" the context

- Sequence transduction problems
- Machine translation *(e.g. English to Spanish)*
- Image captioning

… and so on

Transformer: Background

- For machine translation and transduction problems we use a category of models called *seq2seq* (sequence to sequence) models



Transformer: Background

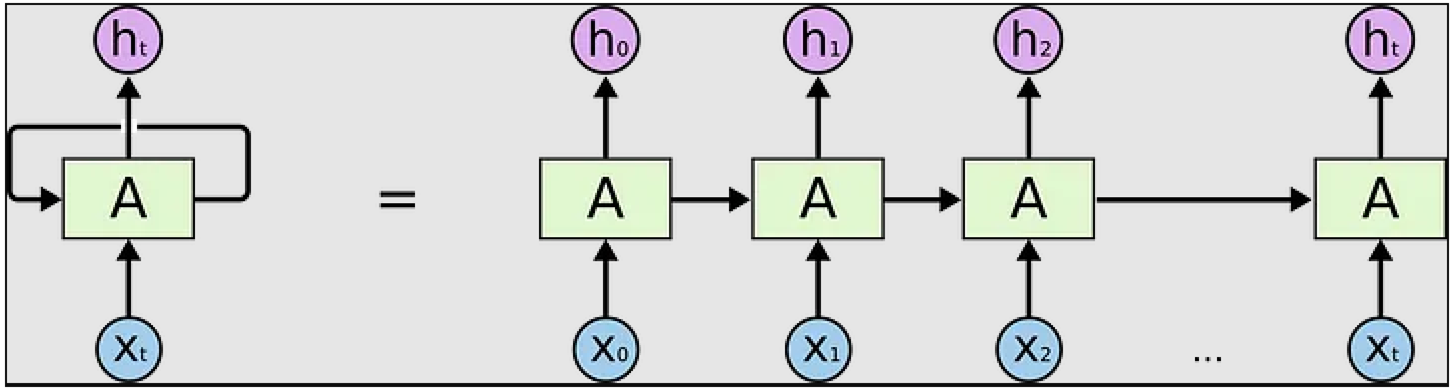
- To correctly translate an input sequence into an output sequence we *need to maintain the context*
- And to maintain the context the neural network *needs to understand the dependencies* between two words
- But to understand dependencies the model *needs to remember the past information* (previous input)

Recurrent Neural Network (RNN)

- Standard neural network architectures cannot *“remember”* previous inputs (as much)
- That is where **Recurrent Neural Networks *(RNNs)*** come into play
- The ***RNN*** architecture have loops to allow information to persist between inputs

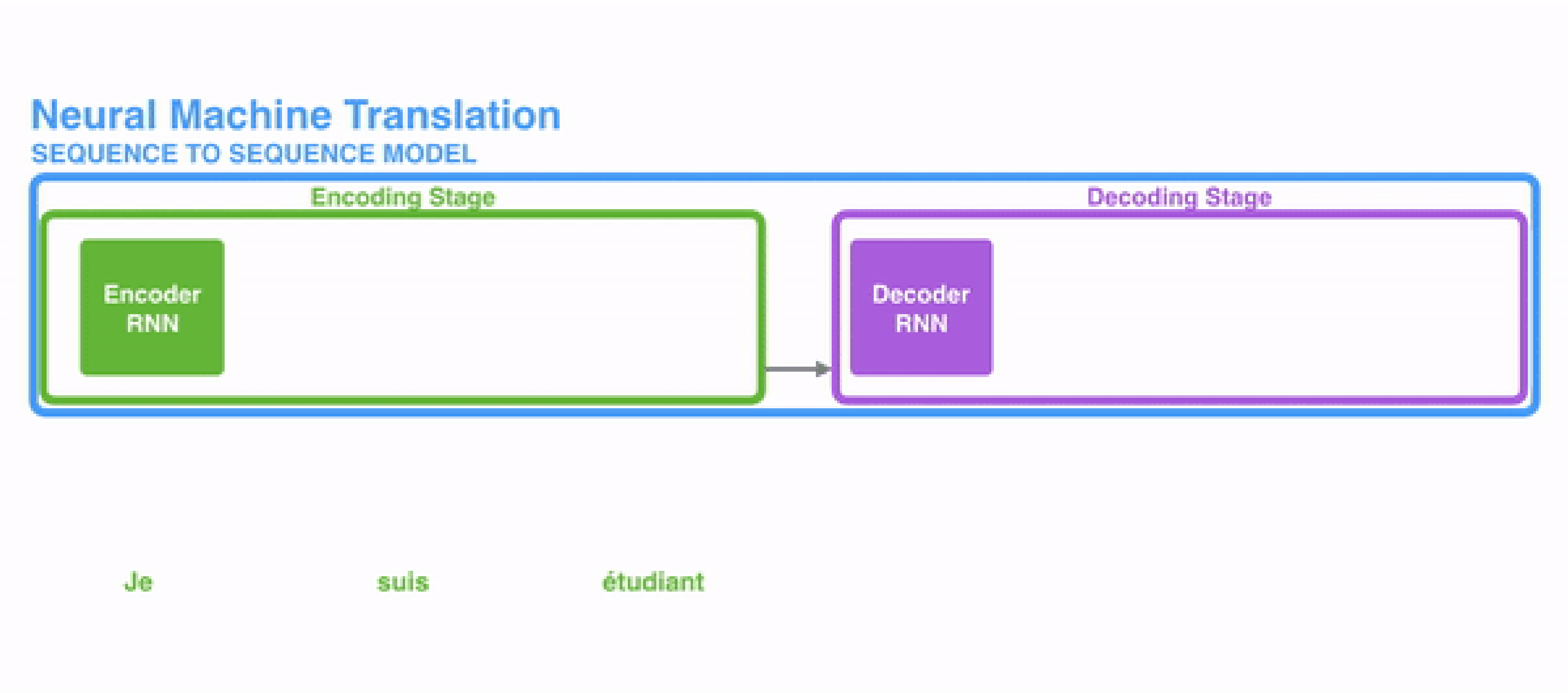
Recurrent Neural Network (RNN)

- Each word is set as an input in the **RNN** architecture
- The word is ***encoded*** as the output of the hidden layer and is fed to the next layer along with another input word in the sequence



Recurrent Neural Network (RNN)

- The output from the final encoder is passed to the second part of the network to get ***decoded*** into the target sequence



Recurrent Neural Network (RNN)

- Some major drawbacks of **RNN** are:
  - Slow training
  - They cannot grasp long-term dependencies between words

- **Example:**
  *“ **Dressrosa Bank** offers various account types to the customers. But the most prominent one is saving account at this **bank** ”*

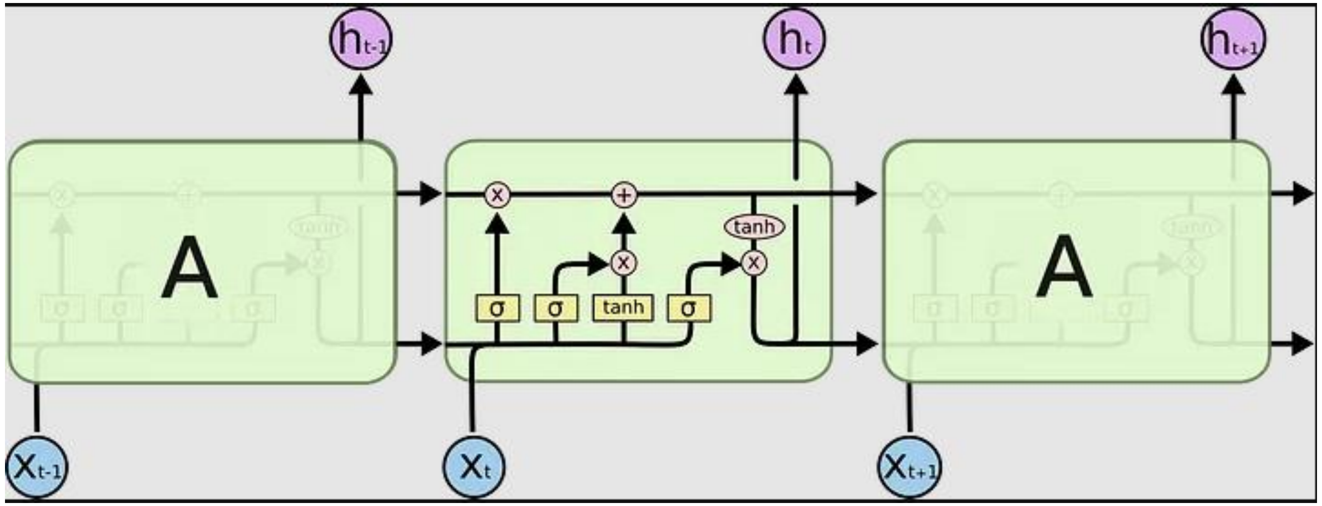
RNN here:
“Ha. I know this bank”

RNN here:
“Uh?!? What bank???”

RNN with LSTM (Long Short Term Memory)

- The **LSTM** model solves the long-term dependency problem in RNNs *(somewhat!)*

- The **LSTM** allows some input information to *"float across"* the encoder layers



RNN with LSTM (Long Short Term Memory)

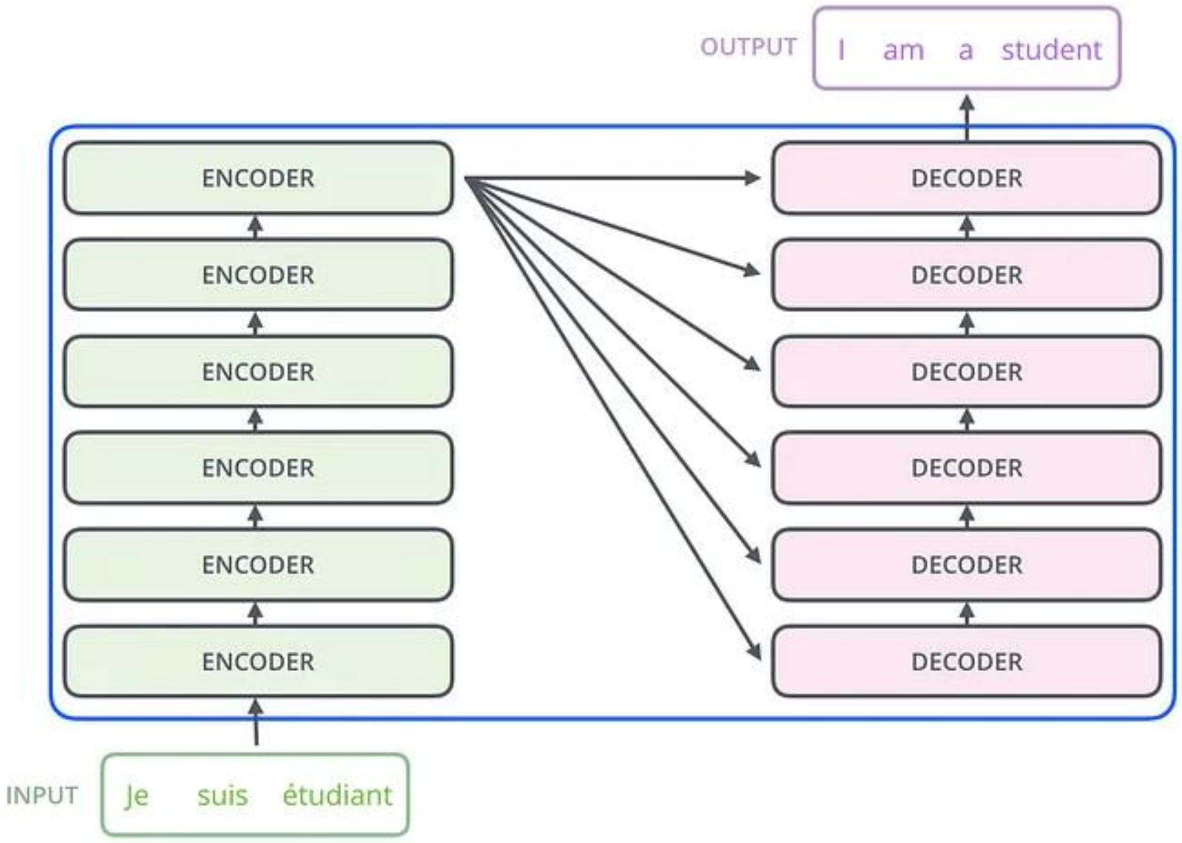
- The drawbacks of **LSTM** architecture is:
  - Lack of parallelization
  - Slow training
  - Can grasp long-term dependencies… *until they become too long-term*

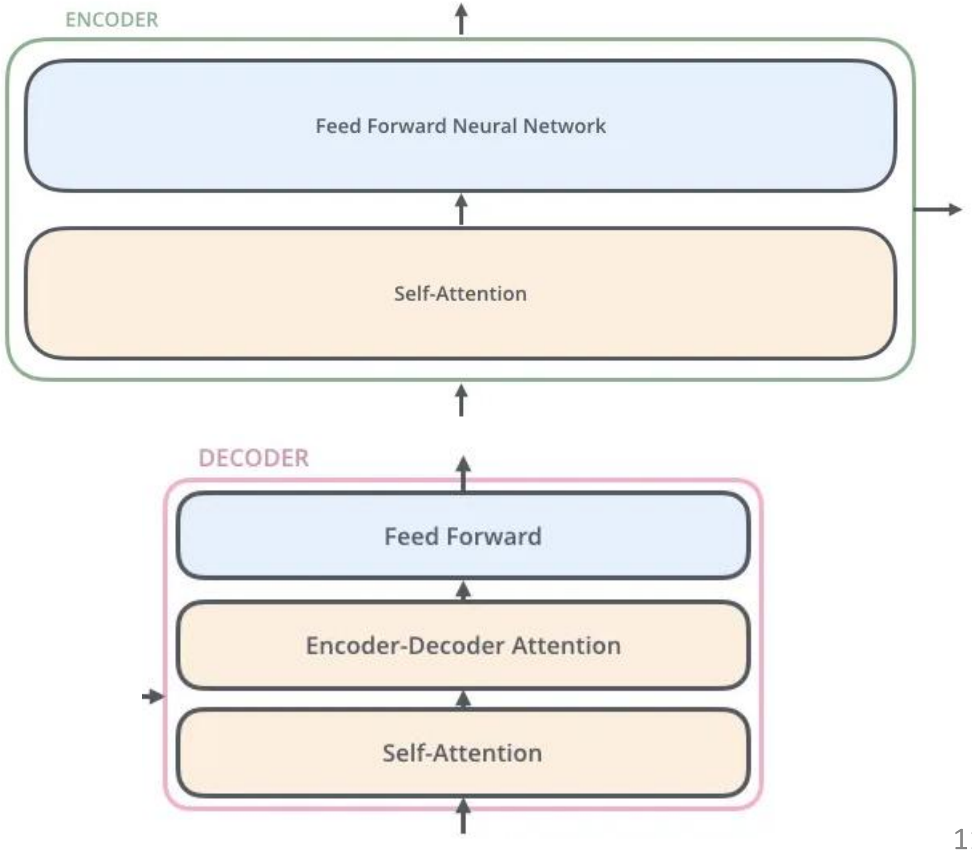
- To overcome the shortcomings of RNN and LSTM we need ***attention***

- **Attention** is a technique that allows input word information to be passed all the way to the decoders

Transformer: Architecture

- A **transformer** uses self-attention and boosts the training speed by allowing parallelization





Transformer: In Recommender Systems

- At times the users do not directly ask for a product recommendation

- Collaborative filtering requires historical interactions, content-based filtering requires user's rating profile, neural recommendations require both item and user feature embedding

- What if the recommendation is to be made without any of these?
  i.e. translating user activity or dialogue to product recommendation

Transformer: In Recommender Systems

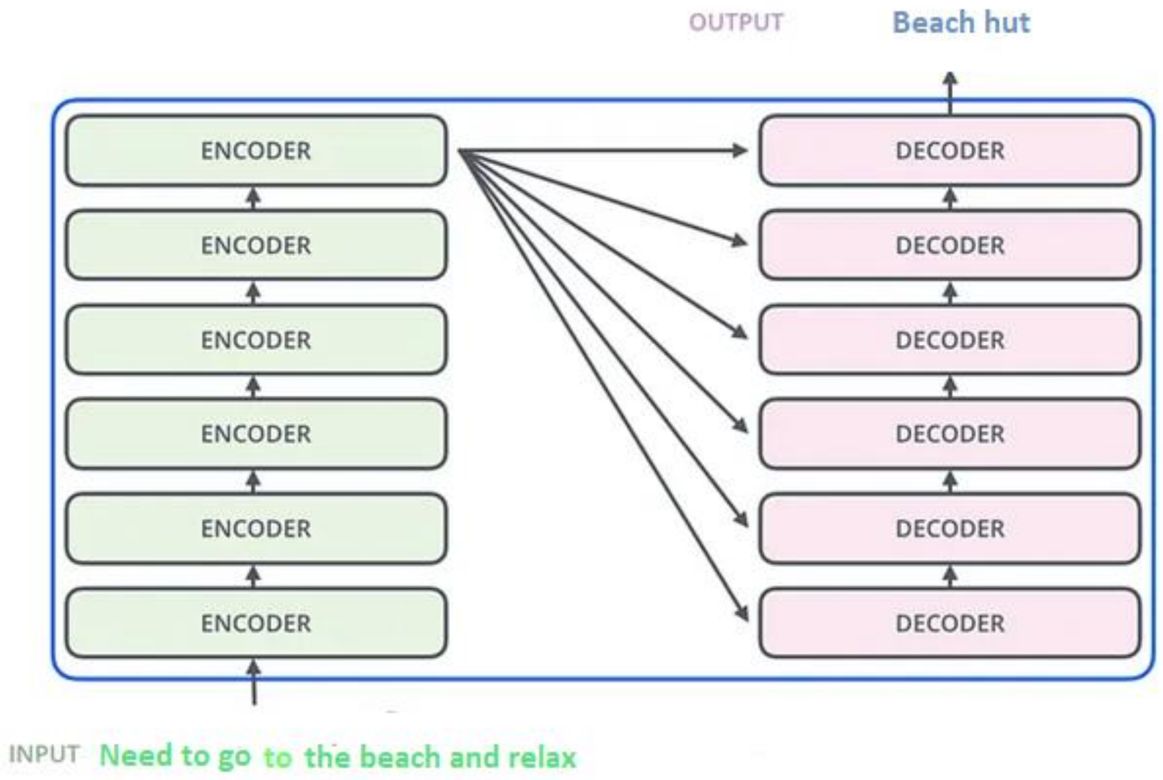
- At times the users do not directly ask for a product recommendation

- Transformers can grasp the context from user’s dialogue and translate it to a vector of relevant products

- Acquiring the input sequence from user involves both technical and ethical considerations
  e.g. should we use the microphone of user’s mobile to gather the relevant context

Transformer: In Recommender Systems

- Transformers can grasp the context from user’s dialogue and translate it to a vector of relevant products



Future Direction

- Dialogue-based recommendations
- Deduction-based recommendations
- Generative recommendations