**Detailed Notes on Sequence-to-Sequence Modeling**

**Introduction:**

* In the previous class, we discussed **advanced RNN methods** like **LSTM** and **GRU**.
* These architectures help solve the **vanishing gradient problem**.
* Today, we look at **sequence-to-sequence (seq2seq)** modeling which became prominent around **2015–2016**.

**What is a Sequence-to-Sequence Problem?**

* Input: A sequence
* Output: A sequence

**Examples:**

* Machine Translation (e.g., Hindi → English)
* Summarization (Long doc → short doc)
* Dialogue systems, speech generation, music generation
* Image captioning (Image → text)

**Types:**

* Many-to-Many: Translation, Dialogue
* Many-to-One: Classification (can be considered seq2seq depending on mapping)
* One-to-Many: Image Captioning, Music/Speech Generation

**Basic Seq2Seq Architecture:**

* Uses **two RNNs**:
  + **Encoder RNN**: Processes input sequence
  + **Decoder RNN**: Generates output sequence

**Flow:**

1. Encoder processes full input sequence and produces a final hidden state.
2. This hidden state is passed to the decoder as initial state.
3. Decoder begins generating output tokens one at a time.

**Teacher Forcing:**

* During training, instead of using the predicted token at each step, we feed the **actual ground truth token**.
* This helps the model **train faster and more accurately**.
* **Note:** During inference, teacher forcing is **not** used.

**Loss Computation:**

* Compute **loss at each output time step**.
* Total Loss = Sum of individual losses over time steps / number of tokens.
* **End-to-End training** using **Backpropagation Through Time (BPTT)**.

**Conditional Language Modeling View:**

* The seq2seq model is a **conditional language model**.
* Conditioned on the **input sequence X**, the model generates each output token:

P(y1∣X),P(y2∣X,y1),…,P(yT∣X,y1,...,yT−1)P(y\_1 | X), P(y\_2 | X, y\_1), \dots, P(y\_T | X, y\_1, ..., y\_{T-1})

**Training vs Inference:**

* **Training:** Uses teacher forcing. Learns embedding + RNN weights + projection weights.
* **Inference:** Uses decoder to generate token by token.

**Weight Matrices:**

* Encoder: W (hidden), W\_e (embedding)
* Decoder: W', W\_e', U (projection to vocab size)
* **Embeddings differ** if input and output languages differ (e.g., Hindi to English).

**Decoding Strategies:**

**1. Greedy Decoding:**

* At each time step, pick the token with **maximum probability**.
* **Drawback**: If a mistake is made early, it can't be corrected (no backtracking).

**2. Exhaustive Search:**

* Consider **all possible token combinations**.
* Impractical due to exponential complexity: VTV^T where V=V = vocabulary size, T=T = time steps.

**3. Beam Search:**

* Compromise between greedy and exhaustive.
* Maintain top **K hypotheses** (branches) at each step instead of all.
* Complexity: KTK^T, where K≪VK \ll V

**Terminology:**

* **Hypothesis**: A possible output sequence.
* Stop generation when **End-of-Sequence (EOS)** token is generated.
* Choose the hypothesis with the **highest total probability score**.

**Summary:**

* Seq2Seq models are powerful for tasks where **both input and output are sequences**.
* Encoder-Decoder structure enables **handling variable-length sequences**.
* Training uses **teacher forcing and BPTT**.
* Inference uses strategies like **greedy decoding** or **beam search**.
* Beam search provides a tradeoff between **accuracy and complexity**.