Sequence-to-Sequence Models

Tianxiang (Adam) Gao

October 28, 2024

Outline

Recap: Recurrent Neural Networks

- **Audio Waveform**: A 1D array represents the amplitude of the sound over time, *e.g.*, 16kHz
- **One-Hot Encoding**: Each word in a **vocabulary** is a **binary** one-hot vector.
- **Challenges in Text Data**: Curse of dimensionality and long-run dependencies.
- **Language Models**: Assigns probabilities to a given sequence of words
- **Neural Language Model**: Model the probability distribution of the next word given the history:

$$\mathbb{P}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t) = f_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t).$$

- **RNNs**: Encode the history into a hidden state $\boldsymbol{h}_t$ updated by combing with the current word $\boldsymbol{x}_t$:

$$\boldsymbol{h}_t = \tanh(\boldsymbol{W}_h h_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b}_h), \quad \text{and} \quad \hat{\boldsymbol{y}}_t = \text{softmax}(\boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y)$$

- **Training RNNs**: backpropagation through time
  - Forward (simplified): $\boldsymbol{h}_t = \phi(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t)$
  - Backward (simplified): $d\boldsymbol{h}_t = \boldsymbol{W}_h^\top \left(\boldsymbol{\phi}'_{t+1} \odot d\boldsymbol{h}_{t+1}\right) + \boldsymbol{W}_y^\top \left(\boldsymbol{\sigma}'_t \odot d\boldsymbol{y}_t\right)$
- **Generation**: Sample the next word from the predicted probability distribution produced by RNNs.
- **RNN Types**: One-to-many, many-to-one, or many-to-many structures for different tasks.
- **Vanishing/Exploding Gradients**: $\boldsymbol{h}_t = \mathcal{O}\left(a^t\right)$ and $d\boldsymbol{h}_t = \mathcal{O}\left(b^{T-t}\right)$.

Recap: Recurrent Neural Networks

- **Gated Recurrent Unit (GRU)**: Gates helps maintain long-term dependencies:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{W}_x \boldsymbol{x}_t), \quad \text{and} \quad \boldsymbol{h}_t = \boldsymbol{z}_t \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t$$

- **Long Short-Term Memory (LSTM)**: Use a cell state $\boldsymbol{c}_t$ to maintain long-term dependencies.

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \quad \text{and} \quad \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)$$

- **Bidirecitonal RNNs**: The concatenated hidden state: $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t]$
- **Deeper RNNs**: Each layer ℓ computes its hidden state using the hidden state from the layer $\ell - 1$:

$$\boldsymbol{h}_t^{\ell} = \tanh(\boldsymbol{W}_h^{(\ell)} \boldsymbol{h}_{t-1}^{(\ell} + \boldsymbol{W}_x^{(\ell)} \boldsymbol{h}_t{}^{(\ell-1)})$$

- **Drawbacks of One-Hot Representation**: orthogonality and high dimensionality
- **Word Embedding**: Words are represented as **dense vectors** in a **lower-dimensional space**.

$$\boldsymbol{e} = \boldsymbol{E}\boldsymbol{x}$$

- **Continuous Bag of Words (CBOW)**: Predicts the target word given the context
- **Skip-Gram**: Predicts the context words given a target word.
- **Negative Sampling**: Reformulates the context-target predictions as a **binary** classification:

$$\mathcal{L}(\boldsymbol{E}) = -\sum_{(\boldsymbol{e}_c, \boldsymbol{e}_t)} \log \sigma(\boldsymbol{e}_t^\top \boldsymbol{e}_c) - \sum_{(\boldsymbol{e}_c, \tilde{\boldsymbol{e}}_t)} \log \sigma(-\boldsymbol{e}_t^\top \tilde{\boldsymbol{e}}_c),$$

where $\tilde{e}_t$ is **negative** target samples outside the context window.

Outline

Word Analogy Task

- **Objective**: Assess the quality of word embeddings by testing how well they capture **semantic** and **syntactic** relationships between words.
- **Goal**: Given an analogy of the form: *"a is to b as c is to ???"*, find the word d that completes the analogy correctly.
- **Examples**:
  - Semantic analogy: *"Paris is to France as Washington, D.C. is to the United States."*
  - Syntactic analogy: *"Run is to Running as Swim is to Swimming."*
- **Vector Arithmetic**: $\boldsymbol{e}_d \approx \boldsymbol{e}_b - \boldsymbol{e}_a + \boldsymbol{e}_c$, *i.e.*, *linear space*
- The word d is chosen as the vector $\boldsymbol{e}_d$ closest to $\boldsymbol{e}_b - \boldsymbol{e}_a + \boldsymbol{e}_c$ based on *cosine similarity*:

$$d = \arg\max_{x \in V} S_C(\boldsymbol{e}_x, \boldsymbol{e}_b - \boldsymbol{e}_a + \boldsymbol{e}_c)$$

  where V is the vocabulary, and

$$S_C(\boldsymbol{u}, \boldsymbol{v}) = \cos(\theta) = \frac{\boldsymbol{u}^\top \boldsymbol{v}}{\|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|}$$

  where θ is the angle between vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.
- **Evaluation**: The predicted word d is evaluated by comparing it to the correct answer from the analogy dataset.

GloVe: Global Vectors for Word Representation

- **Objective**: Create a word embedding model that captures both local context and **global** statistical information from a text corpus.
- **Co-occurrence Matrix**:
  - $\boldsymbol{X}_{ij}$: Number of times word j appears in the context of word i.
  - $\boldsymbol{X}_i = \sum_j \boldsymbol{X}_{ij}$: Total occurrences of any word in the context of word i.
  - $\mathbb{P}(\boldsymbol{w}_j \mid \boldsymbol{w}_i) = \boldsymbol{X}_{ij}/\boldsymbol{X}_i$: Probability of word j occurs in the context of word i.
- **Word Comparison in Context**:
  - Compare words $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ in the context $\tilde{\boldsymbol{e}}_k$ using a **probability ratio**:

$$\exp\left\{(\boldsymbol{e}_i - \boldsymbol{e}_j)^\top \tilde{\boldsymbol{e}}_k\right\} = \frac{\mathbb{P}(\boldsymbol{w}_i \mid \boldsymbol{w}_k)}{\mathbb{P}(\boldsymbol{w}_j \mid \boldsymbol{w}_k)} = \frac{X_{ki}}{X_{kj}}$$

  - Taking the log yields:

$$\boldsymbol{e}_i^\top \tilde{\boldsymbol{e}}_k - \boldsymbol{e}_j^\top \tilde{\boldsymbol{e}}_k = \log X_{ki} - \log X_{kj} \quad \Rightarrow \quad \boldsymbol{e}_i^\top \tilde{\boldsymbol{e}}_k \sim \log X_{ki}$$

- **Cost Function**:
  - The GloVe model learns word vectors $\boldsymbol{e}_i$ and context vectors $\tilde{\boldsymbol{e}}_k$ by minimizing:

$$J = \sum_{i,k} f(X_{ik}) \left(\boldsymbol{e}_i^\top \tilde{\boldsymbol{e}}_k + b_i + \tilde{b}_k - \log(X_{ik})\right)^2$$

  - $f(X_{ik})$ is a weighting function for co-occurrences, while b_i and $\tilde{b}_k$ are bias to maintain symmetry.

Pennington, J., Socher, R., & Manning, C. D. (2014). "GloVe: Global Vectors for Word Representation", EMNLP 2014

Bias in Word Embeddings

- **Problem**: Word embeddings trained on large datasets often encode societal biases, like gender stereotypes:

  "Man is to Computer Programmer as Woman is to Homemaker?"

- **Identifying Bias Direction**: Compute the average difference between **definitional pairs**:

$$\begin{cases} \vec{e}_{\text{man}} - \vec{e}_{\text{woman}} \\ \vec{e}_{\text{he}} - \vec{e}_{\text{she}} \\ \dots \end{cases} \quad \Rightarrow \quad \vec{e}_{\text{bias}} = \frac{1}{N} \sum_{i=1}^{N} (\vec{e}_{x_i} - \vec{e}_{y_i})$$

  This average can be replaced by advanced techniques like PCA.

- **Neutralization**: Project non-definitional words onto the space orthogonal to the bias direction to remove bias:

$$\vec{e} \leftarrow \vec{e} - \frac{\vec{e}^{\top} \vec{e}_{\text{bias}}}{\|\vec{e}_{\text{bias}}\|^2} \cdot \vec{e}_{\text{bias}}$$

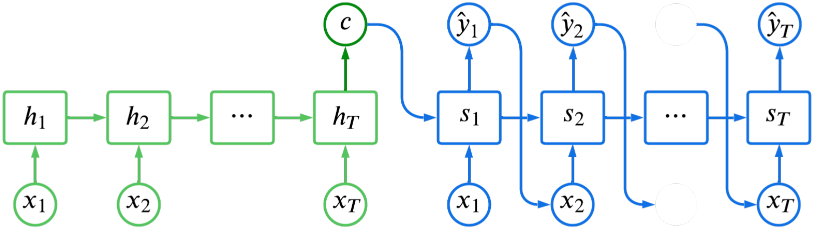
- **Equalizing Pairs**: Adjust equalize pairs (like "brother" and "sister") to have equal and opposite projections along the gender direction, making them **equidistant** from the gender-neutral axis.
- **Identifying Gendered Words**: Train a classifier to distinguish between gender-specific and neutral words using a set of definitional pairs.

Bolukbasi et al., "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," NeurIPS 2016.

Outline

Seq2Seq: Sequence-to-Sequence Models

- **Define**: Sequence-to-Sequence (Seq2Seq) models are designed to handle tasks where both input and output are sequences of variable length, *e.g.*, machine translation or summarization.
- **Example**: "Jane visite l'Afrique en septembre." $\Longrightarrow$ "Jane visits Africa in September."
- **Encoder-Decoder Architecture**
  - **Encoder**: Processes and compresses the input sequence into a fixed-length **context vector**.
  - **Decoder**: Uses the context vector to generate the output sequence sequentially



Here c is the context vector summarizing the *entire* input sequence.

Cho, et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." EMNLP 2014.

Conditional Language Model

- **Language Model**: Assigns probabilities to a sequence of words $\{\boldsymbol{x}_t\} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$:

$$\mathbb{P}(\{\boldsymbol{x}_t\}) = \mathbb{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \prod_{t=1}^{T} \mathbb{P}(\boldsymbol{x}_t \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1})$$

  where each conditional probability is modeled as:

$$\mathbb{P}(\boldsymbol{x}_t \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1})$$

- **Conditional Language Model**: Assigns probabilities to a target sequence $\{\boldsymbol{y}_t\}$ *given* an input sequence $\{\boldsymbol{x}_t\}$:

$$\mathbb{P}(\{\boldsymbol{y}_t\} \mid \{\boldsymbol{x}_t\}) = \prod_{t=1}^{T'} \mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$$

  where the conditional probability for each word in the target sequence is:

$$\mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = g_{\boldsymbol{\phi}}(\boldsymbol{y}_{t-1}, \boldsymbol{s}_t, \boldsymbol{c})$$

  with the entire input sequence stored in the **context vector** $\boldsymbol{c}$.

Cho, et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." EMNLP 2014.

Beam Search

Greedy Search: Selects the most probable word at each step, which may lead to suboptimal sequences.
Beam Search: Tracks **multiple** high-probability sequences simultaneously to improve overall accuracy.

- **Initialization**: Start with the *seed* token and choose the top k words based on the probability distribution.
- **Expansion**: Predict the next word for each candidate, generating new sequences.
- **Pruning**: Keep the top k sequences with the highest log-probability scores, discarding the rest.

$$\mathbb{P}(\boldsymbol{y}_2, \boldsymbol{y}_1 \mid \boldsymbol{c}) = \mathbb{P}(\boldsymbol{y}_2 \mid \boldsymbol{c}, \boldsymbol{y}_1) \cdot \mathbb{P}(\boldsymbol{y}_1 \mid \boldsymbol{c})$$

- **Repeat**: Continue expanding and pruning until an *end-of-sequence* token is generated or max length is reached.

Remark

- **Beam Width (k)**: Requires k identical decoders to update candidate sequences simultaneously.
- **Advantages**: Balances between accuracy and computation; larger k increases accuracy but demands more computational resources.

Sutskever et al., "Sequence to Sequence Learning with Neural Networks," NeurIPS 2014.

Numerical Stability and Error Analysis

Log-Probabilities for Stability:

- Since $\mathbb{P}(\cdot) \in [0, 1]$, the product of probabilities can approach zero, causing numerical instability.
- To address this, log-probabilities are used:

$$\begin{aligned} \boldsymbol{y}^* &= \underset{\boldsymbol{y}}{\operatorname{argmax}}\, \mathbb{P}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T'} \mid \boldsymbol{c}) \\ &= \underset{\boldsymbol{y}}{\operatorname{argmax}}\, \frac{1}{T'} \sum_{t=1}^{T'} \log \mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{c}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) \end{aligned}$$

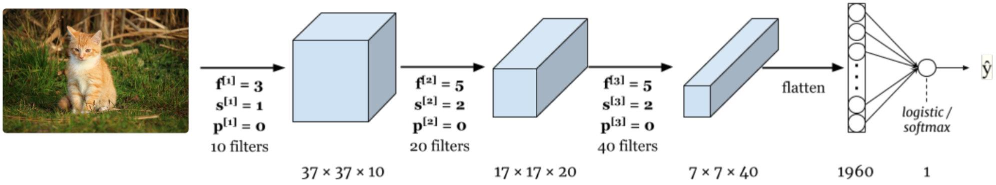
This transformation helps prevent underflow and enables stable computation of probabilities.

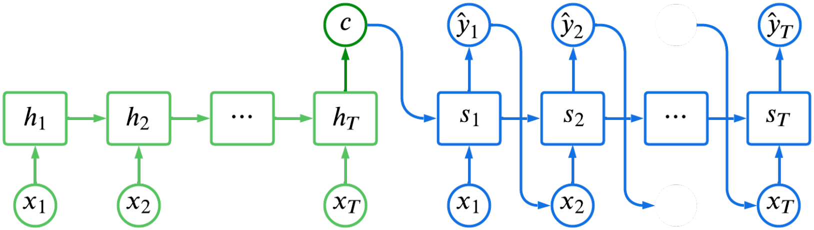
Error Analysis in Beam Search:

- Let $\boldsymbol{y}^*$ be the optimal sequence and $\hat{\boldsymbol{y}}$ the model's predicted sequence.
- **If** $\mathbb{P}(\boldsymbol{y}^* \mid \boldsymbol{c}) > \mathbb{P}(\hat{\boldsymbol{y}} \mid \boldsymbol{c})$: Increasing beam width can improve accuracy by exploring more potential sequences.
- **If** $\mathbb{P}(\boldsymbol{y}^* \mid \boldsymbol{c}) \leq \mathbb{P}(\hat{\boldsymbol{y}} \mid \boldsymbol{c})$: Increasing beam width won't help, as errors stem from model limitations.

Sutskever et al., "Sequence to Sequence Learning with Neural Networks," NeurIPS 2014.

Image Captioning





"A small orange kitten sits attentively on green grass, surrounded by natural, dried foliage in the background, giving a calm and serene outdoor setting."

Vinyals, et al. "Show and tell: A neural image caption generator." CVPR 2015.

BLEU Score: Bilingual Evaluation Understudy

BLEU Score: Bilingual Evaluation Understudy (BLEU) evaluates a machine-generated **candidate translation** $\hat{y}$ by comparing it to a *list* of reference translations $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_M\}$.

- **Candidate Translation**: "the cat the cat sat"
- **Reference Translation 1**: "the cat sat on the mat"
- **Reference Translation 2**: "the cat sat by the mat"

Precision: Measures how many n-grams in the candidate match the reference translations.

- **Candidate Bigrams**: {"the cat", "cat the", "the cat", "cat sat"}
- **Reference 1 Bigrams**: {"the cat", "cat sat", "sat on", "on the", "the mat"}
- **Reference 2 Bigrams**: {"the cat", "cat sat", "sat by", "by the", "the mat"}

$$\text{Precision} = \frac{\text{Number of matching n-grams}}{\text{Total n-grams in candidate}} = \frac{3}{4} \implies \textbf{Inflate the score!}$$

Papineni, et al. "BLEU: a method for automatic evaluation of machine translation." ACL 2002.

Modified Precision

Modified Precision: Limits the count of an n-gram to the maximum it appears in any reference.

- **Count in Candidate**:
  - "the cat" appears **2 times** in the candidate.
  - "cat the" appears **1 time** in the candidate.
  - "cat sat" appears **1 time** in the candidate.
- **Clipped Count**:
  - "the cat" appears at most **1 time** in any reference.
  - "cat sat" appears at most **1 time** in any reference.

$$\text{Modified Precision} = \frac{\text{Clipped number of matching n-grams}}{\text{Total n-grams in candidate}} = \frac{2}{4}$$

BLEU Score Formula: Uses modified precision p_n for n-grams:

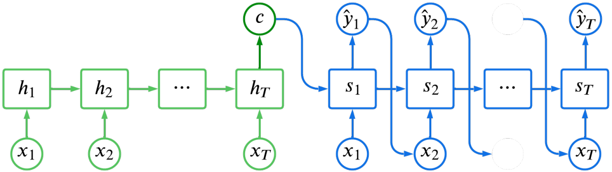
$$\text{BLEU} = BP \cdot \exp\left(\frac{1}{N}\sum_{n=1}^{N}\log p_n\right)$$

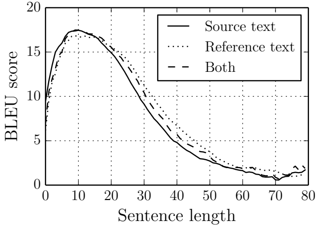
- N: Maximum n-gram length considered (typically $N = 4$).
- **Brevity Penalty (BP)**:

$$BP = \begin{cases} \exp\left(1 - \frac{r}{c}\right), & \text{if } c < r \\ 1, & \text{if } c \geq r \end{cases}$$

where c is the length of the candidate translation and r is the length of the reference translation.

Limitations of RNN Encoder-Decoder Framework





- **Encoder**: Process and compress the input sequence $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ into a context vector $\boldsymbol{c}$.

$$\boldsymbol{h}_t = f_{\boldsymbol{\theta}}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t), \qquad \boldsymbol{c} = \boldsymbol{h}_T.$$

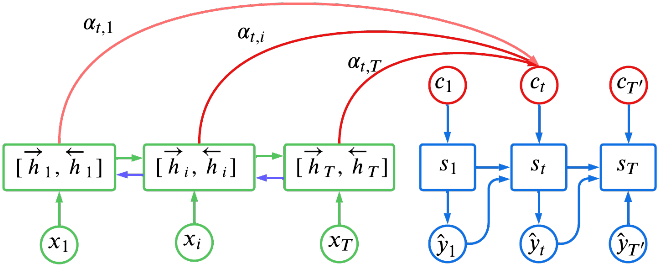
- **Decoder**: Uses the context vector $\boldsymbol{c}$ to generate the output sequence sequentially

$$\mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_{t-1}) = \mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{s}_t), \quad \text{where} \quad \boldsymbol{s}_t = g_{\phi}(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c})$$

Note: Encoding all information into a single vector $\boldsymbol{c}$ may cause information loss for longer sequences.

Bahdanau, et al. "Neural machine translation by jointly learning to align and translate." ICLR 2015

Distinct Context Vector in Attention Mechanism



- **Distinct Context Vector for Each Target Word**: Each target word $\boldsymbol{y}_t$ has a unique context vector $\boldsymbol{c}_t$, allowing the model to focus on relevant input parts.

$$\mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) = \mathbb{P}(\boldsymbol{y}_t \mid \boldsymbol{s}_t), \quad \text{where} \quad \boldsymbol{s}_t = g_\phi(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t)$$

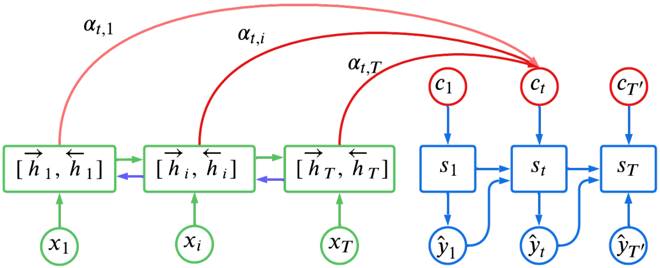
- **Context Vector Computation**: The context vector $\boldsymbol{c}_t$ is computed as a weighted sum of encoder hidden states $\boldsymbol{h}_i$, tailored to the current decoding step.

$$\boldsymbol{c}_t = \sum_{i=1}^{T} \alpha_{t,i} \boldsymbol{h}_i$$

where **attention weights** $\alpha_{t,i}$ indicate the relevance of each hidden state $\boldsymbol{h}_i$ for generating $\boldsymbol{y}_t$.

Bahdanau et al., "Neural Machine Translation by Jointly Learning to Align and Translate," ICLR 2015

Attention Mechanism in Seq2Seq



- **Attention Weights**: Computed from an **alignment score** $\boldsymbol{e}_{t,i}$ between the decoder's previous hidden state $\boldsymbol{s}_{t-1}$ and each encoder hidden state $\boldsymbol{h}_i$.

$$\alpha_{t,i} = \text{softmax}(\boldsymbol{e}_{t,i}), \quad \text{where} \quad \boldsymbol{e}_{t,i} = a(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)$$

- **Alignment Model**: The alignment function a is a feed-forward neural network, trained jointly with Seq2Seq to optimize attention.
- **Bidirectional Encoder**: The encoder uses a bidirectional RNN to capture both past and future context, enhancing comprehension of each input word's meaning.

Bahdanau et al., "Neural Machine Translation by Jointly Learning to Align and Translate," ICLR 2015