NLP Course 2025

Breaking the fixed-length barrier in neural sequence modeling

Previous Weeks:

- Week 1: Statistical language models
- Week 2: Neural language models
- Week 3: RNN/LSTM architectures

This Week:

- Sequence-to-sequence architecture
- Encoder-decoder models
- Attention mechanisms
- Applications and impact

Learning Objectives:

- Understand variable-length sequence processing
- Master encoder-decoder architecture
- Grasp attention mechanism fundamentals
- Apply seq2seq to real problems

Key Innovation:
Decoupling input and output sequence lengths

Foundation for modern NLP: machine translation, summarization, dialogue

Fixed-Length Limitation:

- Traditional RNNs: one input $\rightarrow$ one output
- Cannot handle length mismatches
- Real tasks need flexibility

Real-World Examples:

- "I love you" (3) $\rightarrow$ "Je t'aime" (2)
- "Thank you" (2) $\rightarrow$ "Arigato" (1)
- Article (500 words) $\rightarrow$ Summary (50)

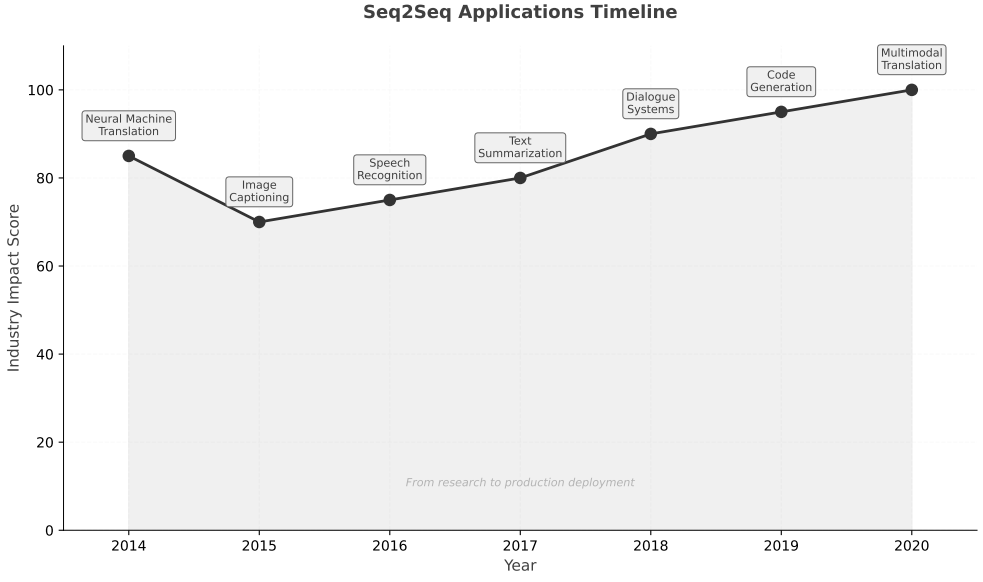
Failed Approaches:

- Padding to maximum length
- Truncating sequences
- Forced 1:1 word mapping

Requirements:

- Variable input length T
- Variable output length T'
- No fixed relationship $T \neq T'$

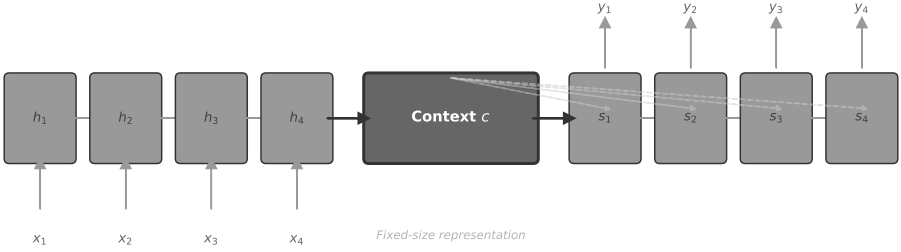
The core challenge that motivated sequence-to-sequence models (2014)


Seq2Seq Applications Timeline
Industry Impact Score
Year
0
20
40
60
80
100
2014
2015
2016
2017
2018
2019
2020
Neural Machine Translation
Image Captioning
Speech Recognition
Text Summarization
Dialogue Systems
Code Generation
Multimodal Translation
From research to production deployment

Variable input length

Variable output length

Sequence-to-Sequence Architecture



Fixed-size representation

Encoder

Decoder

Encoder:

$$h_t = \text{LSTM}(h_{t-1}, x_t)$$
$$t = 1, \ldots, T$$
$$c = h_T$$

Decoder:

$$s_t = \text{LSTM}(s_{t-1}, y_{t-1}, c)$$
$$t = 1, \ldots, T'$$
$$P(y_t \mid y_{<t}, x) = \text{softmax}(W_s s_t + b)$$

Training Objective:

$$\mathcal{L} = -\sum_{t=1}^{T'} \log P(y_t^* \mid y_{<t}^*, x)$$

Key Components:

- h_t: encoder hidden states
- c: context vector
- s_t: decoder hidden states
- y_t: output tokens

Context vector c bridges variable-length sequences through fixed representation

```
def forward(self, src, tgt):    Encode source _, (h, c) = self.encoder(src)
Decode with context outputs = [] for t in range(tgt.shape[0]):  out, (h, c) =
self.decoder( tgt[t], (h, c) ) outputs.append( self.output(out) ) return
torch.stack(outputs)
```

PyTorch implementation showing core architecture components

Key Design Choices:

- LSTM for both encoder/decoder
- Final hidden state as context
- Teacher forcing during training
- Beam search for inference

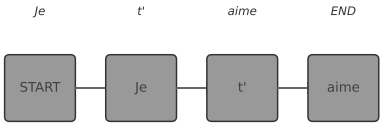
Training Strategy:

- Mini-batch processing
- Gradient clipping
- Learning rate scheduling
- Early stopping on validation

Typical Hyperparameters:

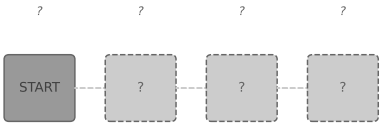
- Hidden size: 256-512
- Layers: 2-4
- Dropout: 0.2-0.3
- Batch size: 32-64

Teacher Forcing



Ground truth used as input

Inference Mode



Previous predictions used as input

Teacher forcing accelerates training but creates exposure bias at inference

The Challenge:

- Entire input compressed to fixed c
- Information loss inevitable
- Longer sequences suffer more
- Context vector becomes bottleneck

Observable Effects:

- Performance degrades with length
- Details lost in translation
- Poor on rare words
- Struggles with long-range dependencies

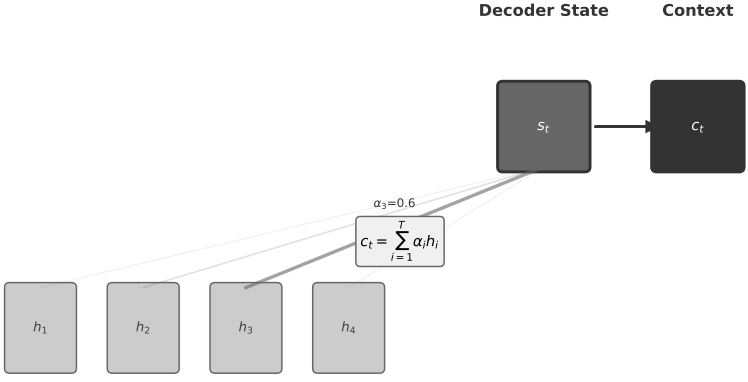
Attempted Solutions:

- Larger hidden dimensions
- Bidirectional encoders
- Multiple layers
- Ensemble methods

The Real Solution:
Attention mechanism (next section)
Impact on Performance:
BLEU drops 1 point per 5 tokens after length 20

Fixed-size bottleneck motivates attention mechanism innovation

Attention Mechanism



Weighted sum of encoder states based on relevance

Attention Computation:

$$e_{tj} = a(s_{t-1}, h_j)$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})}$$

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j$$

Alignment Functions:

- Dot: $a(s, h) = s^T h$
- General: $a(s, h) = s^T W h$
- Concat: $a(s, h) = v^T \tanh(W[s; h])$

Benefits:

- No information bottleneck
- Direct connection to all inputs
- Learns alignment automatically
- Handles long sequences

Interpretation:

- α_{tj}: attention weights
- Soft alignment between sequences
- Differentiable selection mechanism
- Visualizable for debugging

Bahdanau et al. (2015): +5.5 BLEU points on WMT14 English-French

Bahdanau Attention (2015):

- Uses previous decoder state
- Computes attention before output
- More computationally expensive
- Better for alignment tasks

Luong Attention (2015):

- Uses current decoder state
- Computes attention after RNN
- More efficient computation
- Simpler implementation

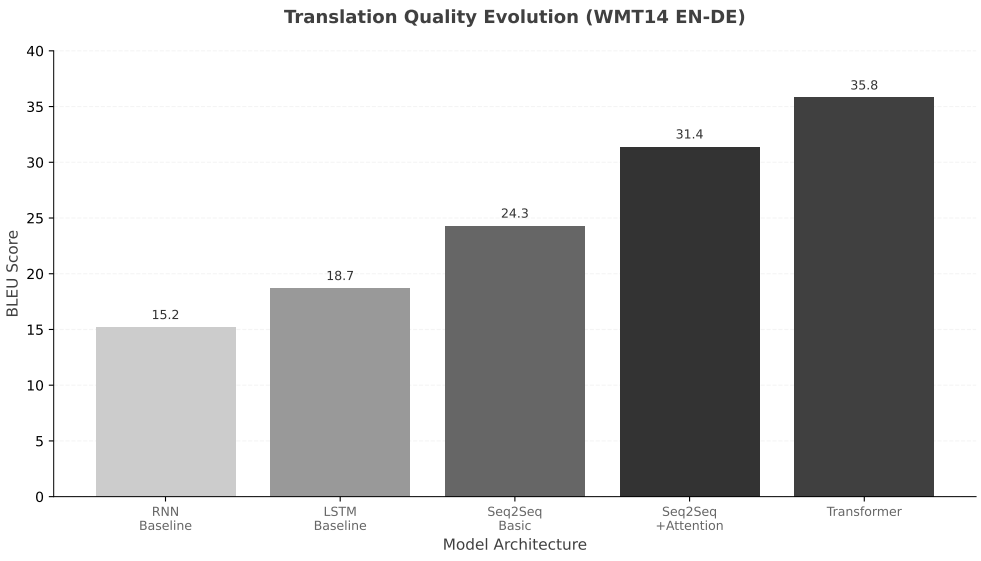
Global vs Local:

- Global: attends to all positions
- Local: window around aligned position
- Monotonic: forward-only attention
- Hard: discrete selection (non-differentiable)

Modern Extensions:

- Multi-head attention
- Self-attention
- Cross-attention
- Scaled dot-product

Different attention mechanisms suit different tasks and constraints


Translation Quality Evolution (WMT14 EN-DE)
BLEU Score
0
5
10
15
20
25
30
35
40
15.2
18.7
24.3
31.4
35.8
RNN Baseline
LSTM Baseline
Seq2Seq Basic
Seq2Seq +Attention
Transformer
Model Architecture
Higher is better. Human performance ~40

BLEU Score:

- N-gram precision based
- Brevity penalty for short outputs
- Range: 0-100 (higher better)
- Standard for MT evaluation

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{4} \frac{1}{4} \log p_n\right)$$

Other Metrics:

- METEOR: considers synonyms
- ROUGE: for summarization
- ChrF: character-level F-score
- BERTScore: semantic similarity

Human Evaluation:

- Fluency: grammatical correctness
- Adequacy: meaning preservation
- Preference: side-by-side comparison
- Error analysis: categorized mistakes

Typical BLEU Scores:

- $<$10: Poor quality
- 10-20: Understandable
- 20-30: Good quality
- 30-40: High quality
- $>$40: Near human

Multiple metrics needed for comprehensive evaluation

Greedy vs Beam Search:

- Greedy: select top-1 at each step
- Beam: maintain top-k hypotheses
- Trade-off: quality vs speed
- Typical beam size: 4-10

Algorithm:

1. Initialize with <START>
2. For each position:
   - Expand all beams
   - Score continuations
   - Keep top-k
3. Return best complete sequence

Length Normalization:

$$\text{score} = \frac{1}{T^{\alpha}} \sum_{t=1}^{T} \log P(y_t)$$

Coverage Penalty:

- Prevents repetition
- Encourages attending to all inputs
- Improves translation coverage

Practical Tips:

- Length penalty $\alpha \approx 0.6$
- Early stopping on <END>
- Ensemble multiple models
- Re-rank with language model

Beam search typically improves BLEU by 1-2 points over greedy

Training Issues:

- Gradient explosion/vanishing
- Slow convergence
- Overfitting on short sequences
- Memory constraints

Solutions:

- Gradient clipping (norm 5-10)
- Learning rate scheduling
- Curriculum learning
- Gradient accumulation

Data Processing:

- Vocabulary size (30k-50k)
- Unknown word handling
- Sequence length limits
- Batch padding strategy

Best Practices:

- Reversing source sequences
- Bidirectional encoder
- Residual connections
- Layer normalization

Proper implementation details crucial for state-of-the-art performance

Google Translate (2016):

- GNMT: Google Neural MT
- 8-layer LSTM seq2seq
- Attention mechanism
- 100+ language pairs
- 1 billion daily translations

Technical Details:

- WordPiece tokenization
- Shared encoder-decoder
- Quantization for mobile
- TPU optimization

Performance Gains:

- 60% error reduction vs phrase-based
- Human parity on news (2018)
- Real-time translation
- Offline capability

Other Systems:

- Facebook: fairseq
- Microsoft: Translator
- Amazon: Translate
- DeepL: Transformer-based

Seq2seq enabled neural machine translation revolution (2014-2017)

Abstractive Summarization:

- Generates new sentences
- Seq2seq with attention
- Pointer-generator networks
- Coverage mechanism

Architecture Adaptations:

- Hierarchical attention
- Copy mechanism
- Content selection
- Sentence rewriting

Applications:

- News summarization
- Research paper abstracts
- Email summaries
- Meeting minutes
- Legal document briefs

Challenges:

- Factual consistency
- Length control
- Multi-document input
- Domain adaptation

Pointer networks solve OOV problem in summarization

Conversational AI:

- Context-aware responses
- Multi-turn dialogue
- Personality consistency
- Task-oriented dialogue

Seq2Seq Extensions:

- Hierarchical encoder
- Memory networks
- Persona embedding
- Emotion modeling

Commercial Systems:

- Customer service bots
- Virtual assistants
- Mental health support
- Educational tutors
- Game NPCs

Key Improvements:

- Context carry-over
- Intent recognition
- Slot filling
- Response diversity

Foundation for modern chatbots before large language models

Seq2Seq Legacy (2014-2017):

- Proved end-to-end learning viable
- Introduced attention mechanism
- Established encoder-decoder paradigm
- Enabled many applications

Limitations Leading to Transformers:

- Sequential processing bottleneck
- Long-range dependency issues
- Training inefficiency
- Limited parallelization

Key Innovations Retained:

- Encoder-decoder architecture
- Attention mechanism (scaled up)
- Teacher forcing training
- Beam search decoding

Transformer Advantages:

- Full parallelization
- Self-attention throughout
- Positional encoding
- Multi-head attention

"Attention is All You Need" (2017) builds directly on seq2seq attention

Foundational Papers:

- Sutskever et al. (2014):
  Sequence to Sequence Learning with Neural Networks
- Cho et al. (2014):
  Learning Phrase Representations using RNN Encoder-Decoder
- Bahdanau et al. (2015):
  Neural MT by Jointly Learning to Align and Translate
- Luong et al. (2015):
  Effective Approaches to Attention-based NMT

Impact Timeline:

- 2014: First seq2seq models
- 2015: Attention mechanism
- 2016: Google Translate deployment
- 2017: Transformer supersedes
- Today: Foundation understood

Citations (2024):

- Sutskever: 20,000+
- Bahdanau: 25,000+
- Fundamental to modern NLP

These papers established the foundation for modern neural NLP

Task: Number Translation

```
vocab_in =
'zero': 0, 'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5
vocab_out = '0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5
Your task: 1. Encode word sequence 2. Generate number sequence 3.
        Handle variable lengths
```

Implementation Steps:

1. Tokenize input sequence
2. Embed tokens
3. Run through encoder LSTM
4. Extract context vector
5. Initialize decoder with context
6. Generate output sequence
7. Convert to numbers

Extensions:

- Add attention mechanism
- Try different languages
- Implement beam search
- Visualize attention weights

Simple task demonstrates core seq2seq concepts without complexity

Architecture Insights:

- Encoder-decoder decouples lengths
- Context vector bridges sequences
- Attention solves bottleneck
- Teacher forcing for training

Practical Lessons:

- Implementation details matter
- Beam search improves quality
- Multiple metrics needed
- Domain adaptation crucial

Historical Significance:

- Enabled neural MT revolution
- Introduced attention concept
- Foundation for Transformers
- Still used in production

Modern Relevance:

- Core concepts remain valid
- Understanding aids Transformer learning
- Useful for constrained devices
- Baseline for new research

Seq2seq: the bridge between RNNs and modern Transformer architectures

Preview of Week 5:

- Self-attention mechanism
- Multi-head attention
- Positional encoding
- Parallel processing
- "Attention is All You Need"

Building on Seq2Seq:

- Attention becomes primary
- No recurrence needed
- Massive parallelization
- Scale to billions of parameters

Preparation:

- Review attention mechanism
- Understand matrix operations
- Read Vaswani et al. (2017)
- Practice with PyTorch

Key Questions:

- Why abandon recurrence?
- How does self-attention work?
- What enables parallelization?
- Why so successful?

- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. NeurIPS.
- Cho, K., et al. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. EMNLP.
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. ICLR.
- Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. EMNLP.
- Wu, Y., et al. (2016). Google's neural machine translation system. arXiv:1609.08144.
- Vaswani, A., et al. (2017). Attention is all you need. NeurIPS.
- See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. ACL.

Comprehensive reading list available on course website