NATURAL LANGUAGE PROCESSING

Sequence Models

Siamese Networks, Neural Machine Translation, Summarization & QA

Agenda

Part 1: Siamese Networks

Duplicate Detection, Triplet Loss

Part 2: Encoder-Decoder

Seq2Seq Architecture

Part 3: Neural Machine Translation

Attention Mechanism, BLEU Score

Part 4: Text Summarization

Extractive vs Abstractive, ROUGE

Part 5: Question Answering

Retriever-Reader, SQuAD

Part 6: Lab and Exercises

Exercise 3, Final Check-in

REVIEW

Recap: Neural Networks for NLP

RNN

$h_t = \tanh(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t)$

Sequential processing with hidden state.
Suffers from vanishing gradient.

LSTM

$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$

Forget, Input, Output gates.
Cell state for long-term memory.

GRU

$h_t = z_t \odot h_{t-1} + (1-z_t) \odot \tilde{h}_t$

Update & Reset gates.
Simpler than LSTM, similar performance.

NER

B-PER I-PER O

BiLSTM-CRF for sequence labeling.
BIO tagging scheme.

Key Concepts

Recurrent connections enable sequence modeling. Gating mechanisms solve long-term dependency issues. Bidirectional processing captures both past and future context. CRF layer ensures valid output sequences.

Siamese Networks

Learning Similarity Between Inputs

Duplicate Detection • One-Shot Learning • Triplet Loss

What are Siamese networks?

Core idea

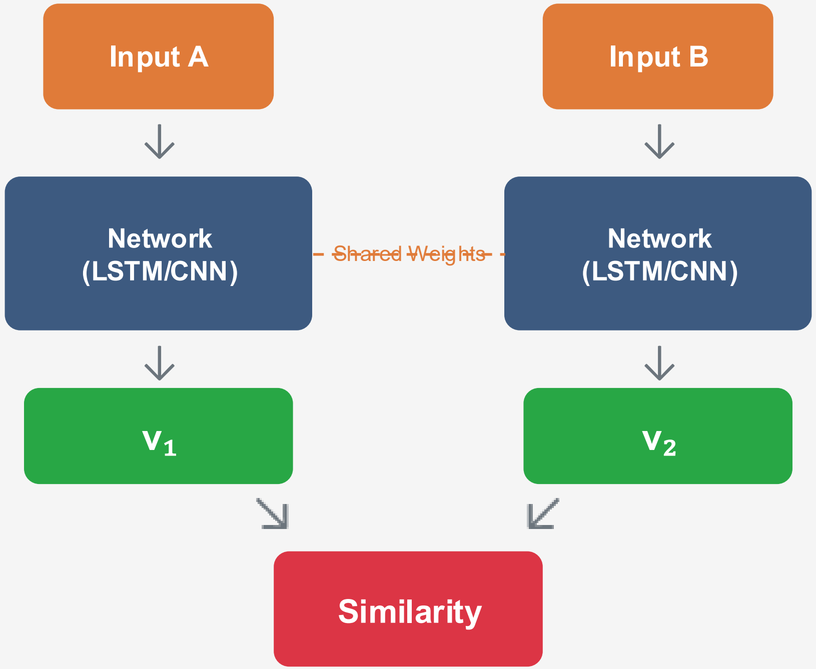
A Siamese network consists of twin networks that share identical weights and architecture, processing two inputs simultaneously to compute their similarity.

Applications

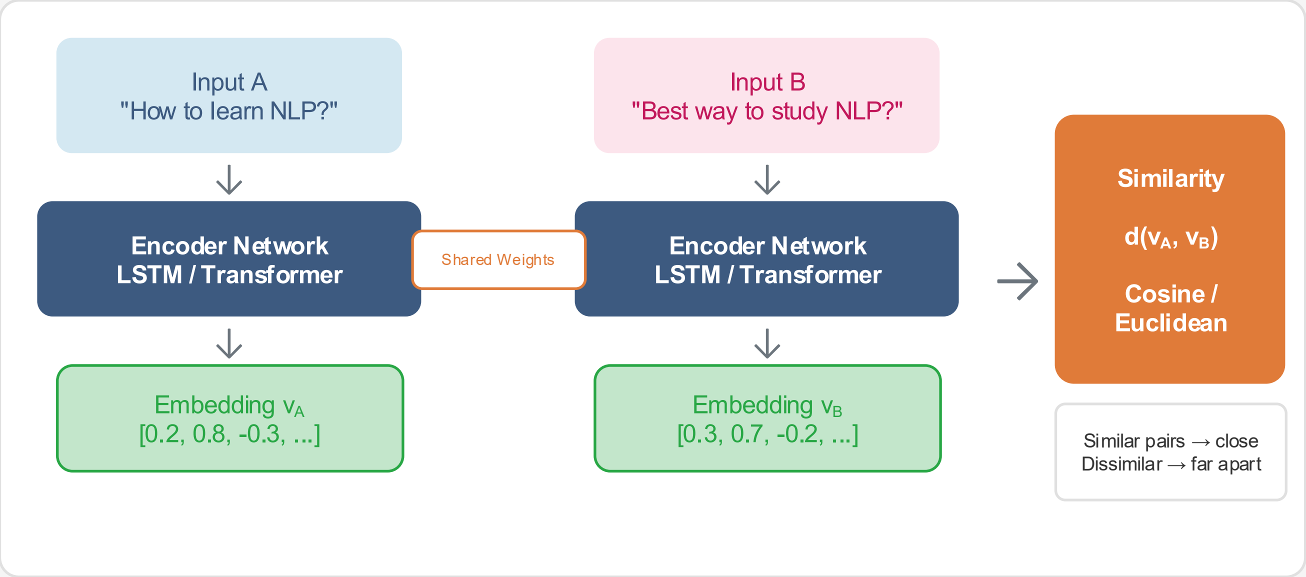
- Duplicate Question Detection
  (Quora, StackOverflow)
- Face Verification & Recognition
- Signature Verification
- One-Shot Learning
- Semantic Text Similarity

Why "Siamese"?

Named after Siamese twins - two networks that are identical and joined, sharing the same parameters. When one learns, the other automatically learns too!



Siamese network architecture



Twin Networks

Identical networks with shared weights.

Distance Metric

Cosine or Euclidean in embedding space.

Learning Goal

Learn to distinguish similar vs dissimilar.

Contrastive loss function

Contrastive loss formula

$$L = \frac{1}{2}\left[(1-y)\cdot\ d^2 + y\cdot\max(0, m-d)^2\right]$$

Where d is Euclidean distance between embeddings, y=0 for similar pairs, y=1 for dissimilar pairs, m is the margin hyperparameter (m=1.0 or 2.0).

Similar (y=0)

Minimize d^2 to pull closer

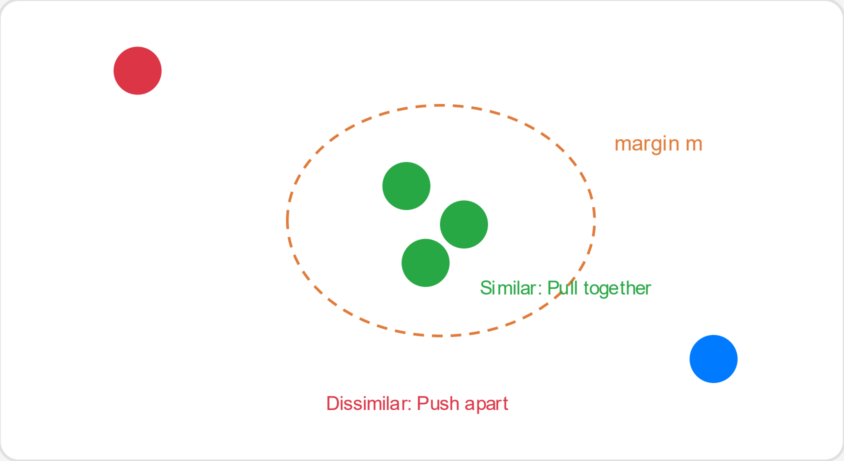
Dissimilar (y=1)

$\max(0, m\text{-}d)^2$ to push apart

Insight

Contrastive loss learns an embedding space where similar pairs are close and dissimilar pairs are separated by at least margin m.

Embedding space visualization



Margin Selection

Too small margin means not enough separation. Too large makes training difficult.

Triplet loss function

Triplet Components

Anchor (A)
Reference sample

Positive (P)
Same class as A

Negative (N)
Different class

Triplet Loss Formula

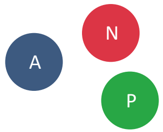
$$L = \max(0, d(A,P) - d(A,N) + \alpha)$$

d(A,P) is distance between Anchor and Positive. d(A,N) is distance between Anchor and Negative. α is the margin (typically 0.2 to 0.5).

Goal d(A,P) + α < d(A,N) → Positive closer than Negative by margin α

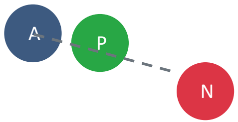
Before vs After Training

Before Training



✗ N closer than P

After Training



✓ P closer than N + α

Hard Triplet Mining

Select triplets where d(A,P) ≈ d(A,N) for faster learning. Easy triplets (d(A,N) >> d(A,P)) contribute 0 loss.

Loss functions comparison

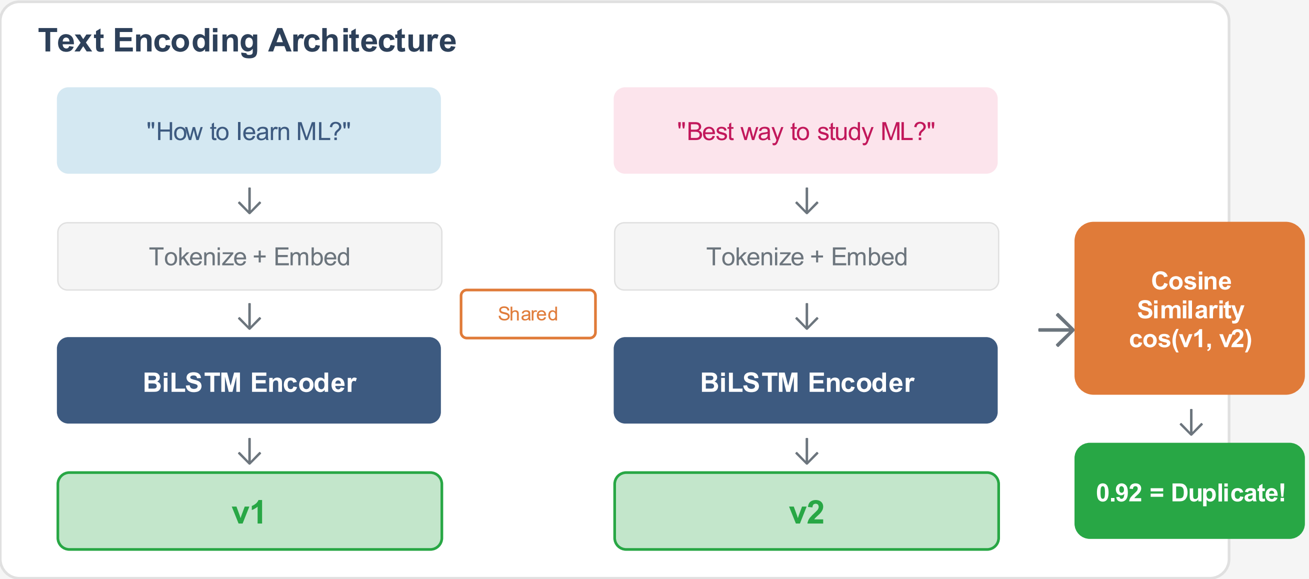
Aspect	Contrastive Loss	Triplet Loss
Input	Pairs (x_1, x_2)	Triplets (Anchor, Positive, Negative)
Label	Binary: y=0 (similar), y=1 (dissimilar)	No explicit labels, implicit from triplet
Formula	$\frac{1}{2} \cdot [(1-y) \cdot d^2 + y \cdot \max(0, m-d)^2]$	$\max(0, d(A,P) - d(A,N) + \alpha)$
Training Complexity	$O(n^2)$ pairs	$O(n^3)$ triplets, needs mining
Pros	Simple, fewer samples needed	Better embeddings, relative distance
Cons	Absolute margin, less flexible	Harder to train, needs mining

When to Use Each Loss

✓ Contrastive: Simple tasks, limited data, verification tasks

✓ Triplet: Fine-grained similarity, ranking, retrieval tasks

Siamese networks for NLP



Sentence Representation

- Mean/Max pooling over states
- Use last hidden state
- Concatenate fwd + bwd

Popular Encoders

- BiLSTM
- Transformer
- BERT, Sentence-BERT

Similarity Metrics

- Cosine similarity
- Euclidean distance
- Manhattan distance

Siamese networks summary

Key Concepts

- **Twin networks with shared weights**
- Learn embedding space for similarity
- Use contrastive or triplet loss
- Inference: compute distance between embeddings
- Effective for few-shot learning scenarios

Loss Functions Comparison

Contrastive Loss	Triplet Loss
Pairs • Simple • $O(n^2)$	Triplets • Complex • $O(n^3)$
Best for: verification	Best for: ranking/retrieval

NLP Applications

Duplicate Detection

Quora question pairs, StackOverflow, customer support deduplication

Semantic Similarity

STS benchmark, paraphrase identification, semantic search

One-Shot Learning

Intent classification with few examples, signature/author verification

Information Retrieval

Dense passage retrieval, document ranking, query-document matching

Encoder-Decoder

Sequence-to-Sequence Architecture

Foundation for Translation, Summarization & Chatbots

Sequence-to-Sequence (Seq2Seq)

What is Seq2Seq?

A neural architecture that transforms an input sequence into an output sequence of potentially different length.

Applications

- **Machine Translation:** EN to VN, EN to FR
- **Text Summarization:** Long doc to short summary
- **Chatbots:** Question to response
- **Speech Recognition:** Audio to text
- **Image Captioning:** Image to description

Insight

Input and output can have different lengths, vocabularies, and even modalities!

Examples of Seq2Seq Tasks

"I love machine learning" → "Tôi yêu học máy"

Machine Translation (EN to VN)

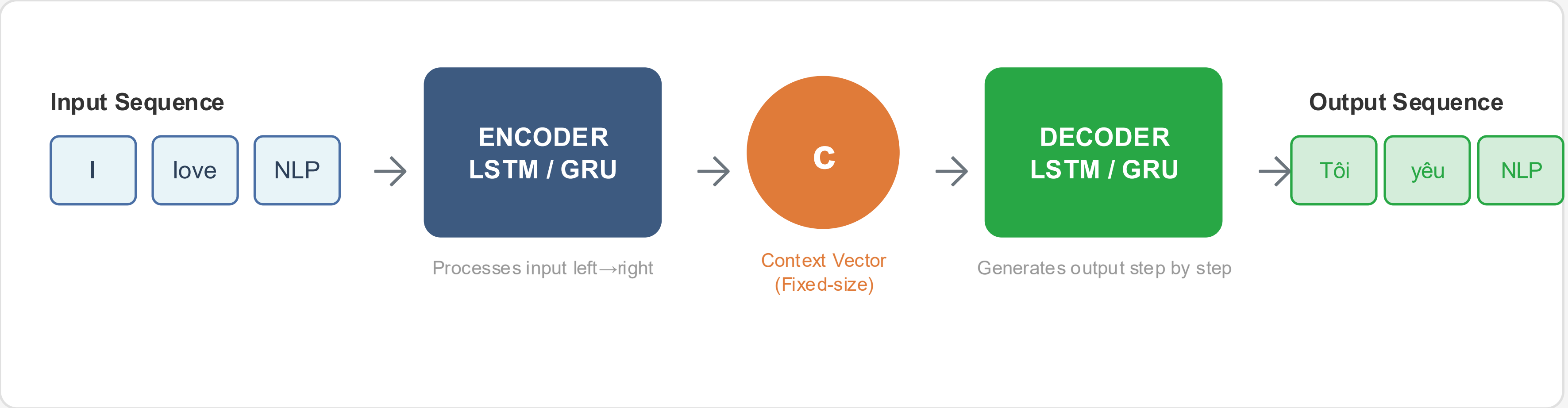
"The quick brown fox jumps over the lazy dog..." → "A fox jumps over a dog."

Text Summarization

"What is the capital of Vietnam?" → "The capital is Hanoi."

Question Answering / Chatbot

Encoder-Decoder architecture



Encoder

Reads input sequence and compresses into fixed-size context vector c = hT (final hidden state)

Context Vector

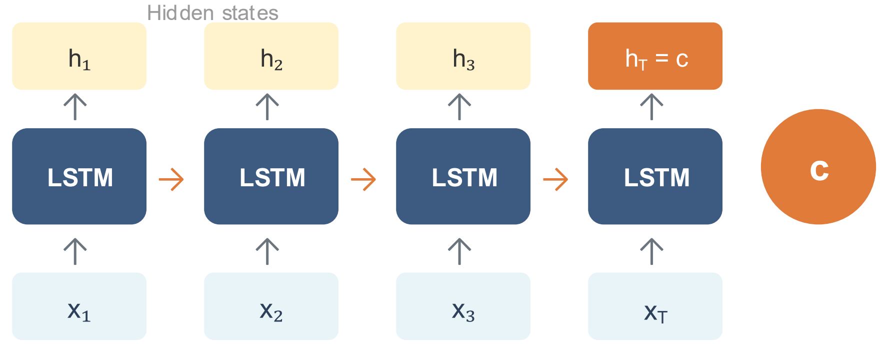
Single vector containing all info about input. Bottleneck for long sequences!

Decoder

Generates output token by token, conditioned on context and previous outputs.

Encoder: processing input sequence

Encoder Unrolled (LSTM/GRU)



Encoder Equations

$h_t = \mathrm{LSTM}(x_t, h_{t-1})$

$c = h_T$ (context vector)

Encoder Options

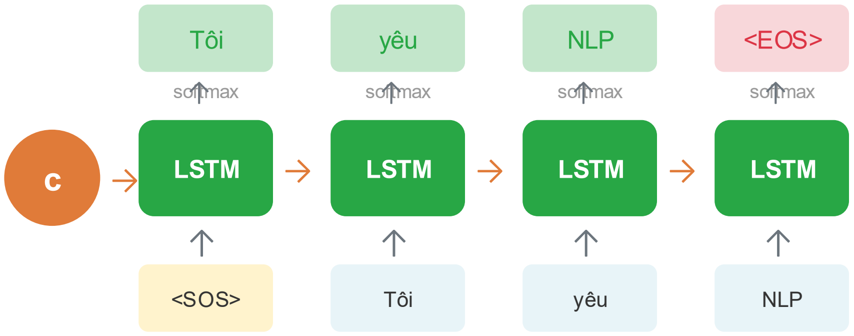
- LSTM / GRU (most common)
- Bidirectional: concat $h^{\rightarrow}$ and $h^{\leftarrow}$
- Multi-layer (stacked)
- Transformer encoder

⚠ **Bottleneck Problem**

All input info must fit in one fixed-size vector c. Hard for long sequences!

Decoder: generating output sequence

Decoder Unrolled (Autoregressive)



Decoder Equations

$$s_t = \mathrm{LSTM}(y_{t-1}, s_{t-1}, c)$$

$$P(y_t) = \mathrm{softmax}(W_o \cdot s_t)$$

Concepts

- **Autoregressive: Each output depends on previous**
- <SOS>: Start of sequence token
- <EOS>: End of sequence token
- Teacher forcing: Use ground truth during training

Inference vs Training

Training: use ground truth y_{t-1}.
Inference: use predicted $\hat{y}_{t-1}$

Training vs Inference

Training (Teacher Forcing)

- **Use ground truth as input to decoder**
- Even if model predicts wrong, use correct token
- Faster convergence
- Parallel computation possible

Loss = $-\sum \log P(y_t^* \mid y<t^*, x)$

```
Input: [<SOS>, "Tôi", "yêu"] → Target: ["Tôi", "yêu", "NLP"]
```

Inference (Autoregressive)

- **Use model's own predictions as next input**
- Sequential generation (cannot parallelize)
- Errors can accumulate (exposure bias)
- Decoding strategies needed

$\hat{y}_t = \text{argmax } P(y \mid \hat{y}<t, x)$

```
<SOS> → "Tôi" → "yêu" → "NLP" → <EOS>
```

Decoding Strategies

Greedy

Pick highest prob token at each step. Fast but suboptimal.

Beam Search

Keep top-k candidates. Better quality, slower.

Sampling

$$P_i = \frac{e^{\frac{\text{logit}_i}{T}}}{\sum e^{\frac{\text{logit}_i}{T}}}$$

Random sampling with temperature. More diverse outputs.

Summary

Key Concepts

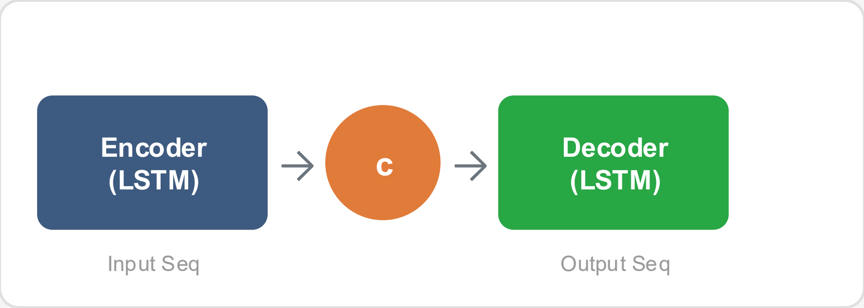
- Encoder compresses input → context vector
- Decoder generates output autoregressively
- Handles variable-length sequences
- Teacher forcing for efficient training
- Foundation for NMT, summarization, chatbots

Limitations

- Bottleneck: All info in one fixed vector
- Long sequences → information loss
- Sequential processing (slow)

→ Solution: Attention Mechanism (Part 3)

Architecture Overview



Applications

- Translation
- Summarization
- Chatbots
- Speech

Neural Machine Translation

Attention Mechanism & BLEU Score

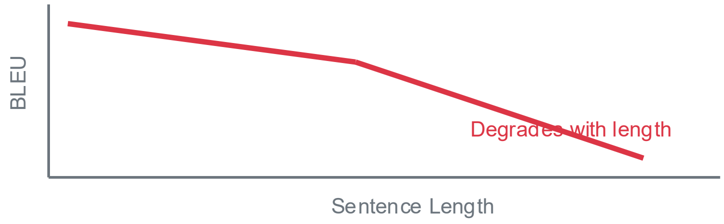
Breaking the Bottleneck with Attention

The bottleneck problem

❌ Problem with Basic Seq2Seq

The entire input sequence is compressed into a single fixed-size vector c. For long sequences, this causes information loss!

Performance Degradation



Key Observation

When translating a word, we don't need ALL input info - just the relevant parts!

✓ Solution: Attention Mechanism

Instead of using one fixed context vector, dynamically focus on different parts of the input at each decoding step.

Intuition: Human Translation

I love machine learning → Tôi yêu học máy

Focus on relevant words

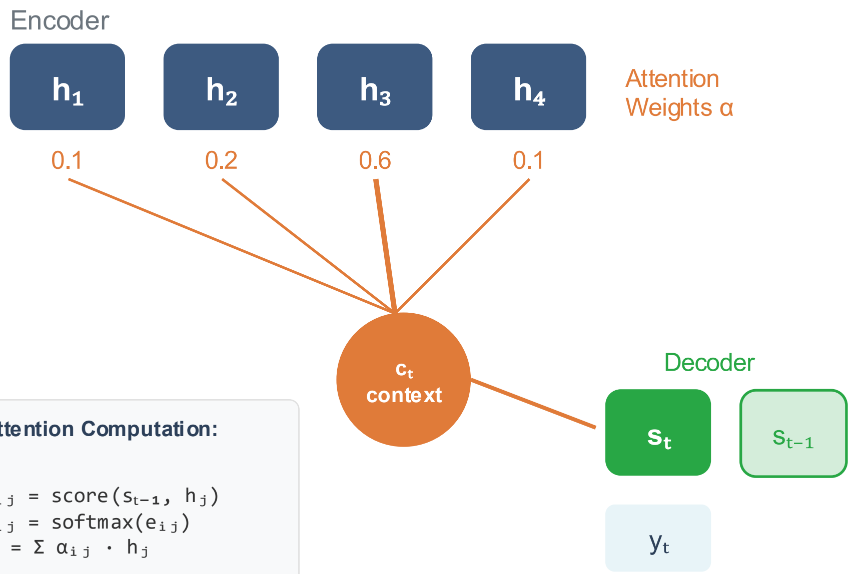
Generate aligned outputs

Paper: Bahdanau et al., 2014

"Neural Machine Translation by Jointly Learning to Align and Translate"

Attention Mechanism

Attention in Seq2Seq



Attention Computation:

$$e_{ij} = \mathrm{score}(s_{t-1}, h_j)$$
$$\alpha_{ij} = \mathrm{softmax}(e_{ij})$$
$$c_t = \Sigma\, \alpha_{ij} \cdot h_j$$

Key Idea

At each decoding step, compute a weighted combination of all encoder hidden states based on their relevance.

Score functions:

- **Dot:** $s^T h$
- **General:** $s^T W_h h$
- **Concat:** $v^T \tanh(W[s;h])$

Benefits

- Handles long sequences
- Provides interpretability
- Learns alignments

Attention Calculation

1

Compute Alignment Scores

$$e_{ij} = \mathrm{score}(s_{t-1}, h_j)$$

Calculate how well each encoder state h_j matches the current decoder state s_{t-1}

2

Normalize with Softmax

$$\alpha_{ij} = \exp(e_{ij}) / \Sigma_k \exp(e_{ik})$$

Convert scores to probability distribution (attention weights sum to 1)

3

Compute Context Vector

$$c_t = \Sigma_j \alpha_{ij} \cdot h_j$$

Weighted sum of encoder hidden states based on attention weights

4

Generate Output

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

Use context vector c_t along with previous state and output to generate next token

Insight: Attention allows the model to "look back" at different parts of the input at each step, solving the bottleneck problem!

Attention Score Functions

Dot-Product Attention

$\text{score}(s, h) = s^\top h$

✓ **Pros:**

- Simple & fast
- No extra parameters
- Works well when dimensions match

✗ **Cons:**

- Requires same dimensions
- May not scale well

Complexity: $O(d)$

General (Multiplicative)

$\text{score}(s, h) = s^\top W_h h$

✓ **Pros:**

- Learnable transformation
- Handles different dimensions
- More flexible

✗ **Cons:**

- Extra parameters W
- Slightly slower

Complexity: $O(d^2)$

Concat (Additive)

$\text{score}(s, h) = v^\top \tanh(W[s;h])$

✓ **Pros:**

- Most flexible
- Non-linear transformation
- Bahdanau original

✗ **Cons:**

- Most parameters
- Slowest computation

Complexity: $O(d^2)$

Recommendation: Scaled Dot-Product (Transformer) divides by $\sqrt{d_k}$ to prevent large values: $\text{score} = (s^\top h) / \sqrt{d_k}$

NMT with Attention: complete architecture

Encoder-Decoder with Attention

Components:

Encoder
BiLSTM

Attention
Bahdanau

Decoder
LSTM + Attn

Training & Performance

Training

- Teacher forcing
- Cross-entropy loss
- Adam optimizer
- Dropout regularization

Performance Gains

- +11 BLEU on long sentences
- Better alignment learning
- Interpretable outputs

Attention visualization

Attention Alignment Matrix (EN → VN)

	I	love	machine	learning
Tôi	0.90	0.05	0.03	0.02
yêu	0.05	0.85	0.05	0.05
học	0.02	0.08	0.45	0.45
máy	0.02	0.03	0.50	0.45

High attention

Low attention

Interpretation

- "Tôi" strongly aligns with "I"
- "yêu" focuses on "love"
- "học máy" attends to both "machine" and "learning"

✓ Benefits of Visualization

- Interpretable alignments
- Debug translation errors
- Understand model behavior
- Detect attention issues

💡 Multi-word Alignment

Some words like "machine learning" map to multiple target words - attention handles this naturally!

BLEU Score Evaluation

BLEU: Bilingual Evaluation Understudy

Automatic metric to evaluate machine translation quality by comparing candidate translation to reference translations.

Brevity Penalty

$$\mathrm{BP} = \min(1, e^{1-\frac{r}{c}})$$

c = candidate length, r = reference length
Penalizes translations shorter than reference

BLEU Formula

$$BLEU = \mathrm{BP} * e^{\sum_{n=1}^{N}} \mathrm{W}_{\mathrm{n}} \log(\mathrm{p}_{\mathrm{n}})$$

p: n-gram precision

BP: Brevity penalty

Score Interpretation

0 1

N-gram Precision

p_n = (# matched n-grams) / (# candidate n-grams)

Advantages

- Fast & automatic
- Correlates with human
- Reproducible

Limitations

- Ignores semantics
- Exact match only
- Needs references

BLEU Score Example

Reference:

"the cat sat on the mat"

Candidate:

"the cat sat on mat"

1 Unigram Precision (p_1)

$p_1 = 5/5 = 1.0$

2 Bigram Precision (p_2)

$p_2 = 3/4 = 0.75$

3 Trigram (p_3)

$2/3 = 0.67$

4 4-gram (p_4)

$1/2 = 0.5$

Brevity Penalty

c = 5 (candidate), r = 6 (reference)
BP = e^(1-6/5) = e^(-0.2) = 0.819

Final BLEU Calculation

$$BLEU = BP \times (p_1 \times p_2 \times p_3 \times p_4)^{(1/4)}$$

$$= 0.819 \times (1.0 \times 0.75 \times 0.67 \times 0.5)^{0.25}$$

BLEU = 0.58

Interpretation

Score 0.58 indicates reasonable quality.
Missing "the" before "mat" reduced bigram+ precision.

Other Evaluation Metrics

ROUGE

Recall-Oriented Understudy for Gisting Evaluation

`ROUGE-N = recall of n-grams`

Use: Summarization tasks

✓ Good for recall-focused tasks

✗ Less common for MT

METEOR

Metric for Evaluation of Translation with Explicit Ordering

`Uses synonyms + stemming`

Use: Translation (handles synonyms)

✓ Better correlation with human

✗ Language-specific resources

BERTScore

BERT-based Semantic Similarity

`Cosine similarity of BERT embeddings`

Use: Any text generation

✓ Captures semantics

✗ Computationally expensive

Human Eval

Manual Human Evaluation

`Fluency + Adequacy ratings`

Use: Gold standard

✓ Most accurate

✗ Expensive, slow, subjective

NMT & Evaluation summary

Key Concepts

- **Attention solves bottleneck problem**
- Dynamic focus on relevant input parts
- Provides interpretable alignments
- Score functions: Dot, General, Concat
- BLEU measures translation quality

BLEU Quick Reference

< 10: Almost useless	10-19: Hard to understand	20-29: Gist is clear
30-40: Understandable	40-50: Good quality	> 50: High quality

Architecture Recap

Encoder → Attention → Decoder

Evaluation Methods

BLEU	ROUGE	METEOR	Human
Automatic	Recall-based	Semantic	Gold std

Text Summarization

Extractive & Abstractive Approaches

Condensing Information with ROUGE Evaluation

Introduction to text summarization

What is Text Summarization?

Automatically creating a shorter version of a document while preserving key information and overall meaning.

Two Main Approaches

Extractive

Select important sentences from the original document

Abstractive

Generate new sentences that capture the meaning

Applications

News digests

Email summaries

Research papers

Meeting notes

Example

Original (200 words):

Researchers at MIT have developed a new AI system that can understand and generate human language with unprecedented accuracy. The system, called GPT-4, uses a transformer architecture... The breakthrough could revolutionize fields from healthcare to education. According to lead researcher Dr. Smith...

Extractive Summary:

Researchers at MIT have developed a new AI system. The breakthrough could revolutionize fields from healthcare to education.

Abstractive Summary:

MIT's new GPT-4 AI system achieves unprecedented language understanding, potentially transforming healthcare and education.

Key Difference

Extractive: Copy-paste sentences
Abstractive: Write new sentences

Extractive summarization

Common Methods

1. TF-IDF Based

Score sentences by sum of TF-IDF weights. Select top-k sentences.

2. TextRank (Graph-based)

Build sentence similarity graph, apply PageRank algorithm.

3. Neural Extractive

BERT + classifier to predict sentence importance.

TextRank Formula

$$S(V_i) = (1 - d) + d \times \sum_{j \in In(i)} \frac{w_{ji}}{\sum_k w_{jk}} \times S(V_j)$$

d = damping factor (0.85), w = similarity weight

Extractive Pipeline

Document → Split Sentences → Score Each → Select Top-K

Advantages

- Grammatically correct
- Factually faithful
- Fast & simple
- No hallucination

Limitations

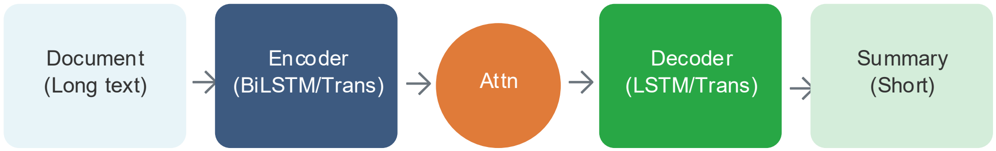
- Redundancy issues
- Lacks coherence
- Can't paraphrase
- Limited compression

Popular Tools

sumy (Python), gensim.summarize, BERT-Extractive, LexRank

Abstractive Summarization

Seq2Seq Architecture for Summarization



Key Techniques

Copy Mechanism

Allows copying rare words from source

Coverage

Prevents repetition in output

State-of-the-Art Models

- BART - Bidirectional + Autoregressive
- T5 - Text-to-Text Transfer
- PEGASUS - Pre-trained for summarization
- GPT/LLMs - Zero-shot capable

Challenges

- **Hallucination** - Generating false info
- **Repetition** - Repeating phrases
- **OOV words** - Out of vocabulary
- **Long documents** - Context limits

Datasets: CNN/DailyMail, XSum, Gigaword, Reddit TIFU

ROUGE Evaluation

ROUGE - Recall-Oriented Understudy for Gisting Evaluation

Measures overlap between generated summary and reference summaries. Standard metric for summarization evaluation.

ROUGE Variants

ROUGE-N	N-gram overlap (ROUGE-1, ROUGE-2 most common)
ROUGE-L	Longest Common Subsequence (word order)
ROUGE-S	Skip-bigram co-occurrence (allows gaps)

ROUGE-N Formula

Recall = Count_match(n-gram) / Count(n-gram in Ref)

F1 = 2 × Precision × Recall / (Precision + Recall)

Example Calculation

Reference:

"the cat sat on the mat"

Generated:

"the cat was on the mat"

ROUGE-1 (Unigram):

Matching: the, cat, on, mat = 5 words. Recall = 5/6 = 0.83

ROUGE vs BLEU

ROUGE	BLEU
Recall-focused	Precision-focused
Summarization	Translation

Typical ROUGE Scores (News)

~44	~21	~40
ROUGE-1	ROUGE-2	ROUGE-L

Text Summarization Summary

Extractive vs Abstractive

Extractive

- Selects existing sentences
- Fast and simple
- No hallucination
- Methods: TF-IDF, TextRank, BERT
- Best for: Factual accuracy needed

Abstractive

- Generates new sentences
- More fluent and concise
- Risk of hallucination
- Models: BART, T5, PEGASUS, GPT
- Best for: Natural, human-like summaries

ROUGE Quick Reference

- ROUGE-1: Unigram overlap
- ROUGE-2: Bigram overlap
- ROUGE-L: Longest common subseq

Coming Next: Question Answering

Apply Seq2Seq to build QA systems and chatbots!

Key: Choose method based on task requirements - accuracy vs fluency!

Question Answering

Retriever-Reader Architecture & SQuAD

Retriever

Reader (MRC)

Datasets

What is Question Answering?

Definition

Question Answering (QA) is the task of automatically answering questions posed in natural language, given a context or knowledge source.

Types of QA Systems

Extractive	Extract answer span from passage
Abstractive	Generate answer in natural language
Open-Domain	Answer from large corpus (Wikipedia)
Closed-Domain	Answer from specific domain documents

Example (Extractive QA)

Context: "The Eiffel Tower was built in 1889 for the World's Fair."

Question: "When was the Eiffel Tower built?"

Answer: 1889

Real-World Applications

- 🔍 Search Engines
- 🤖 Virtual Assistants
- 📞 Customer Support
- 🏥 Medical QA
- 📚 Educational QA

Open-Domain QA Pipeline

Question → Retriever
Find passages → Reader
Extract answer → Answer

Key Insight

Modern QA systems combine Information Retrieval (finding relevant documents) with Machine Reading Comprehension (understanding and extracting answers).

Retriever-Reader Architecture

Question "Who founded Microsoft?" → **RETRIEVER** BM25 / DPR → **Wikipedia** 5M docs → **READER** BERT / RoBERTa → **Answer** "Bill Gates"

RETRIEVER: Find top-k passages

READER: Extract answer

Retriever Methods

Sparse: TF-IDF, BM25

Keyword matching, fast but limited semantics

Dense: DPR (Dense Passage Retrieval)

BERT embeddings, semantic matching

DPR Formula: $sim(q, p) = E_Q(q)^T \times E_P(p)$

Reader (MRC Model)

Input Format

`[CLS] question [SEP] passage [SEP]`

Output

Start and End positions of answer span

$P_{start}(i) = softmax(W_s \times h_i)$

Machine Reading Comprehension

BERT for Extractive QA

Pstart | ... | Pend | ...

↑ Linear + Softmax

BERT / RoBERTa

Transformer Encoder

[CLS] | Who | founded | ... | [SEP] | Bill | Gates | ... | [SEP]

← Question → ← Passage →

Training Objective

$$L = -\log P_{start}(s^*) - \log P_{end}(e^*)$$

Cross-entropy loss for ground truth start (s*) and end (e*) positions

Span Selection Example

Q: Who founded Microsoft?

P: Microsoft was founded by Bill Gates and Paul Allen in 1975.

Start: pos 5
"Bill"

End: pos 6
"Gates"

Evaluation Metrics

Exact Match (EM)

1 if predicted == ground truth, else 0

F1 Score

Popular MRC Models

BERT, RoBERTa, ALBERT, XLNet, ELECTRA, DeBERTa, Longformer

SQuAD Dataset & Benchmarks

SQuAD 1.1

Stanford Question Answering Dataset

- 100K+ question-answer pairs
- Wikipedia passages as context
- All questions are answerable
- Human: 82.3 EM / 91.2 F1

SQuAD 2.0

+ Unanswerable Questions

- 150K+ question-answer pairs
- 50K unanswerable questions
- Model must detect "no answer"
- Human: 86.8 EM / 89.5 F1

Other QA Datasets

Natural Questions
Google search queries

TriviaQA
Trivia questions

HotpotQA
Multi-hop reasoning

CoQA
Conversational QA

SQuAD 2.0 Leaderboard (Top Models)

Human: 86.8/89.5	ALBERT: 90.9/93.2	DeBERTa: 91.4/93.7	GPT-4: 92+/94+

Question Answering Summary

Key Concepts

- **Retriever-Reader Architecture**
- Sparse (BM25) vs Dense (DPR) retrieval
- BERT-based MRC models
- Span extraction for extractive QA
- EM and F1 evaluation metrics

Architecture Recap

Question → Retriever → Reader → Answer

BM25/DPR for retrieval, BERT for reading comprehension

Vietnamese QA Challenges

- Word segmentation required
- Limited training data
- Cross-lingual transfer from mBERT
- Datasets: UIT-ViQuAD, ViNewsQA

Applications

Search	Chatbots
FAQ Systems	Knowledge Base

Coming Next: Lab & Summary

Hands-on practice with Seq2Seq models:

- Build simple chatbot with attention
- Implement QA with Hugging Face

Key Takeaways

Siamese Networks

Twin networks with shared weights. Learn embedding space for similarity. Contrastive/Triplet loss.

Encoder-Decoder

Encoder compresses to context. Decoder generates autoregressively. Bottleneck problem with fixed context.

Attention Mechanism

Dynamic context per step. Weighted sum of encoder states. Handles long sequences, interpretable alignments.

NMT and Summarization

BLEU for translation, ROUGE for summarization. Extractive vs abstractive. Copy mechanism for OOV.

Question Answering

Retriever-Reader pipeline. Sparse vs Dense retrieval. BERT-based MRC, EM/F1 metrics.

Big Picture

These architectures form the foundation for modern NLP. From similarity learning to sequence generation with attention - building blocks for Transformer

References & Further Reading

Papers

Seq2Seq Learning
Sutskever et al. (2014)
"Sequence to Sequence Learning with Neural Networks" - NeurIPS

Attention Mechanism
Bahdanau et al. (2014)
"Neural Machine Translation by Jointly Learning to Align and Translate" - ICLR

Siamese Networks
Koch et al. (2015)
"Siamese Neural Networks for One-shot Image Recognition" - ICML Workshop

BERT for QA
Devlin et al. (2019)
"BERT: Pre-training of Deep Bidirectional Transformers" - NAACL

Online Resources

- Stanford CS224N: Lecture 8-10
- The Illustrated Transformer (Jay Alammar)
- HuggingFace Transformers Course
- SQuAD 2.0 Leaderboard & Dataset

Next Session Preview

Attention Mechanisms & Transformer Architecture - Self-attention, multi-head attention, and the foundation of modern LLMs.

Thank you

Next: Attention Mechanisms & Transformer Architecture