ARTIFICIAL INTELLIGENCE AND NEURAL NETWORKS

Lecture 7b – Attention, Transformers, and Text Generation

Chizhi Chris ZHANG

zhangchizhi@ciomp.ac.cn

Advanced Computing and Digital Technology Research Center

University of Chinese Academy of Sciences

Spring 2026

Today's Question

What we are trying to answer

Why did transformers replace recurrent neural networks as the main engine of modern language models?

Why this lecture matters

AI7 explained what language AI does. NN7 explains why the current neural architecture became strong enough to make large language models practical.

What changes from NN6

NN6 focused on sequence modeling with recurrence and memory through time. NN7 shows what happened when the field stopped passing information step by step and started connecting tokens more directly.

From NN6 to NN7

Last time

We saw how recurrent models read one token after another and carry a hidden state forward.

Today

We ask what happens if a model can look back at many tokens directly instead of squeezing everything through one narrow running state.

One sentence

NN6 was about carrying memory through time. NN7 is about routing information across a sequence more flexibly.

RNNs Were a Good Start

Why RNNs mattered

They made order explicit. Each new token updated the internal state, so sequence data no longer had to be flattened into one fixed bag of inputs.

Why we still start here

Transformers make more sense once you see what problem recurrent models were already trying to solve.

The recap in one sentence

RNNs treated language as a stream. That was a real step forward, but every new token still had to squeeze through one running state.

The Hidden-State Bottleneck

The core bottleneck

Old information has to pass through one limited hidden state before it can influence much later tokens.

What can go wrong

Signals weaken, mix together, or disappear before the model reaches the place where they are actually needed.

This is the memory problem that kept returning in sequence modeling.

A Long Dependency Example

Sentence

“The books on the old shelf near the window ___ missing.”

What the model should connect

The word “books” should still control the later choice “are”, even though many other words sit in between.

Why this is hard

The crucial clue appears early, but the model needs it much later.

Why LSTM Helped

What changed

Gates let the network choose what to keep, what to forget, and what to reveal.

Why this was not the final answer

LSTMs improved memory, but the model was still recurrent and still had to process the sequence step by step.

A simple metaphor

It is like reading with a highlighter, an eraser, and a note card. You decide what to mark, what to discard, and what to pass forward.

Why Sequential Computation Hurts

Recurrent path

token 1 $\rightarrow$ token 2 $\rightarrow$ token 3 $\rightarrow \ldots$

Hardware consequence

Later steps depend on earlier ones, so training cannot fully exploit parallel hardware.

Why industry cared

When model size and dataset size grew, this sequential bottleneck became a serious practical cost.

Why This Became an Industry Problem

Research pressure and product pressure met

Researchers wanted better long-range modeling. Companies wanted faster large-scale training. Both pressures pointed away from strict recurrence.

That is the setup for attention

The field needed a way to connect distant tokens directly while making training much more parallel.

Attention in One Sentence

Core idea

For each token, compute how relevant other tokens are, then mix information using those relevance weights.

Human reading analogy

When reading a sentence, we do not give every earlier word equal importance. We look back selectively.

Why this changed the field

Attention creates a direct path between distant tokens instead of forcing every dependency through one narrow memory channel.

A Small Reading Story

Sentence fragment

“The animal did not cross the street because it was too tired.”

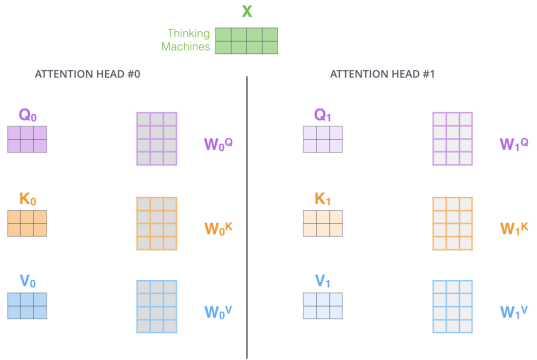
What attention is doing

When the model reads “it”, it should look back strongly toward “animal”, not toward unrelated nearby words.

What changed from RNN thinking

The model does not have to hope the right clue survives inside one hidden state. It can look back more directly.

Query, Key, and Value



Three roles

- query: what the current token is looking for
- key: what each token offers for matching
- value: the content that can be collected

The Smallest Useful Attention Math

Scores and weights

$$s_i = \frac{q^\top k_i}{\sqrt{d_k}}, \qquad \alpha_i = \mathrm{softmax}(s_i)$$

Context vector

$$\text{context} = \sum_i \alpha_i v_i$$

Read in words

Attention is weighted information routing.

A Toy Attention Example

Suppose the model is finding the subject

If one token gets weight 0.62, another gets 0.18, and punctuation gets almost 0, the model is learning where to look.

Why this is intuitive

The model is not averaging the sentence blindly. It is deciding which earlier pieces deserve more influence.

This is why attention feels less mysterious once you read it as selective focus rather than as abstract algebra.

Self-Attention and Cross-Attention

Self-attention

Queries, keys, and values all come from the same sequence.

Cross-attention

Queries come from one sequence while keys and values come from another sequence.

Why both matter

Self-attention helps a sentence understand itself. Cross-attention helps one sequence depend on another, as in translation or retrieval-based setups.

Why Causal Masking Exists

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention Mask →

Masked Scores
(before softmax)

0.11	−inf	−inf	−inf
0.19	0.50	−inf	−inf
0.53	0.98	0.95	−inf
0.81	0.86	0.38	0.90

Generation rule

When predicting the next token, the model must not peek at future tokens on the right.

What masking does

It blocks illegal attention links so the model only uses allowed earlier context.

Why Multi-Head Attention

Single head

One attention pattern may focus mostly on one type of relation.

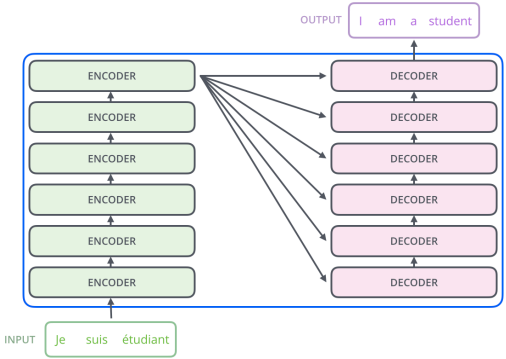
Multiple heads

Different heads can emphasize different patterns at the same time: syntax, agreement, topic, reference, or position.

The practical benefit

The model gets several parallel views of the same sequence instead of betting everything on one attention map.

From Attention to Transformer



Big architectural move

Take attention, add a few stabilizing pieces around it, then stack that block many times.

Why this mattered

The transformer block became the new standard unit for deep sequence modeling.

What One Transformer Block Contains

Main pieces

- multi-head attention
- feed-forward network
- residual connections
- normalization

A simple mental model

Attention lets tokens exchange information. The other pieces keep training stable and give the model extra capacity to reshape what it has learned.

Why Residuals and Normalization Matter

Residual idea

Do not throw away the old representation when adding the new one.

Normalization idea

Keep activations in a healthier range so training behaves more steadily.

Why this page matters

Attention gets most of the attention, but deep models would be much harder to train without these supporting pieces.

Why Position Must Be Added

Subtle issue

Pure attention does not automatically know whether one token came before or after another.

The fix

Add positional information so the model can tell order as well as content.

Without position, “dog bites man” and “man bites dog” would look much more similar than they should.

Encoder, Decoder, and Encoder-Decoder

Encoder	Decoder	Encoder-decoder
Good for reading and representing an input sequence.	Good for generating a sequence step by step with masking.	Good when one sequence must be read before another is generated, as in translation.

A GPT-Style Stack

How to read this diagram

Think of it as the same decoder-style transformer block repeated many times so token representations are refined layer after layer before the model predicts the next-token logits.

Autoregressive Generation

Factorization

$$P(w_{1:T}) = \prod_{t=1}^{T} P(w_t \mid w_{<t})$$

Meaning

The model generates one token at a time, always conditioning on what has already been produced.

Why this is important

This is the mathematical version of the sentence “predict the next token, then repeat”.

Parallelism Changed the Economics

RNN path

Sequence steps must largely wait for one another.

Transformer path

Many token-to-token interactions can be computed in parallel during training.

Why this mattered so much

Better hardware utilization made larger models and larger datasets far more practical.

Training Needed More Than Architecture

Architecture was only one ingredient

- huge datasets
- large compute budgets
- optimization tricks and stable training recipes
- careful data filtering and engineering

Why this belongs in class

Students often hear “transformers changed everything.” That is true, but the change only happened because architecture, data, compute, and systems work grew together.

Training Pipeline View

Pretraining

Learn broad next-token patterns from massive text corpora.

Supervised and preference tuning

Push the model toward more useful responses, better formatting, and behavior people prefer in interaction.

Deployment reality

Serving, filtering, evaluation, caching, and monitoring continue after training. The pipeline does not end when the weights are saved.

From Prompt to Token

Path

prompt → tokenize → embeddings → transformer stack → logits → sampled token

What happens next

The new token is appended to the context, then the process repeats.

Why this page matters

It turns a vague idea of “the model answers” into a concrete loop of repeated prediction.

How Decoding Changes Behavior

Greedy

Always take the highest-probability token.

Temperature

Make outputs more conservative or more diverse.

Top-k / top-p

Restrict sampling to a smaller candidate set.

Why this matters

The same model can sound repetitive, careful, wild, or creative depending on the decoding rule.

Inference Brings New Constraints

Serving-time concerns

- latency
- memory use
- context length
- cost per request
- throughput for many users

Why deployment is a different game

Training asks how to build the model. Inference asks how to serve it fast enough, cheaply enough, and reliably enough.

Why Small Models Still Matter

Bigger is not always better

For some products, a smaller model wins because it is cheaper, faster, easier to run locally, and sometimes easier to control.

Practical rule

Choose the smallest model that meets the task quality you actually need.

Engineering is not a contest for the largest parameter count. It is a search for the best tradeoff.

Why Hallucination Can Happen

Mechanism view

The model is trained to continue text plausibly. Plausibility and factual correctness are related, but they are not identical.

What that means in practice

If the model lacks the needed fact or if the prompt is misleading, it may still produce a fluent answer.

Why this is a transformer lecture issue too

The architecture made scale possible. It did not remove the need for grounding, checking, or evaluation.

Prompting, Fine-Tuning, or Retrieval

Prompting

Use the existing model better by giving clearer instructions and context.

Fine-tuning

Change model behavior by additional training on selected data.

Retrieval

Bring in external knowledge at answer time instead of baking everything into the parameters.

What Teams Actually Evaluate

Before deployment

- task quality
- factual reliability
- refusal and safety behavior
- latency and cost
- robustness on messy real prompts

Why this is the real standard

A model that looks impressive in a benchmark demo may still fail badly in production if these checks are weak.

One Easy Mistake

Misunderstanding

"Transformers solved language understanding."

Better statement

Transformers gave us a much better way to model long-range dependencies and scale language training. They did not make language problems disappear.

Why this distinction matters

Students should separate architectural progress from exaggerated claims about full understanding.

Why Transformers Still Need Judgment

What the architecture gives

Flexible information routing, parallel training, and a strong base for language generation.

What humans still have to do

Decide the task, supply the right context, evaluate the result, and manage risk.

The architecture is powerful. Deployment still depends on human choices.

Why AI7 and NN7 Belong Together

AI7 perspective

Language AI looked like tokenization, embeddings, pretraining, retrieval, and user interaction.

NN7 perspective

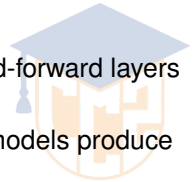
Under that surface, attention and transformer blocks made large-scale language modeling much more effective and practical.

The combined lesson

To understand modern language AI, you need both the application story and the architecture story.

Summary

- RNNs revealed the sequence problem, but recurrence created memory and parallelism bottlenecks.
- Attention lets each token decide which other tokens matter most.
- Transformer blocks combine attention with residuals, normalization, and feed-forward layers to train deep sequence models effectively.
- Causal masking and autoregressive generation explain how decoder-style models produce text one token at a time.
- Modern language systems depend on more than architecture alone: data, compute, tuning, retrieval, evaluation, and serving all matter.



Next Time

Where the course is going

We have now reached the basic language-model stack from two sides: application and architecture.

Next lecture

The next step is to build on this foundation and examine how later topics in the course extend, adapt, or apply these ideas.

Thank You