Text as Data: Transformers

Guest Course – January 2026

Germain Gauthier, Philine Widmer[1]

[1]Bocconi Unversity, Paris School of Economics

USI Lugano

The journey so far

- We've covered increasingly sophisticated text representations:
  - Bag-of-words: Counts, ignores order
  - Embeddings (Word2Vec): Dense vectors, captures semantic similarity
  - Semantic parsing: Extracts linguistic structure
- **Key limitation**: These methods struggle with **context**
  - Word2Vec gives one vector per word, regardless of context
  - "bank" in "river bank" vs. "investment bank"
- **This session**: Transformers
  - Context-aware representations
  - State-of-the-art performance on almost all (supervised) NLP tasks
  - Foundation for modern LLMs (GPT, Claude, etc.)

Before transformers: The sequence modeling problem

Goal: Model sequences where order matters

- *"The Fed raised rates"* $\neq$ *"Rates raised the Fed"*

Traditional approach: Recurrent Neural Networks (RNNs)

- Process text word-by-word, left-to-right
- Maintain "hidden state" that carries information forward
- **Problem 1**: Slow (must process sequentially, can't parallelize)
- **Problem 2**: Struggle with long-range dependencies

Example: *"The Fed, which was established in 1913 and has weathered many crises, **raised** rates."*

Transformer solution: Process entire sequence at once using **attention**

The key innovation: Attention mechanism

Core idea: When processing each word, look at *all* other words and decide which are relevant

Intuition:

"The Federal Reserve raised interest rates."

When processing "raised":

- Pay attention to "Federal Reserve" (who is doing the action?)
- Pay attention to "interest rates" (what is being raised?)
- Ignore "The" (not very informative)

Attention learns these relationships automatically from data

Self-attention: Each word attends to every other word (including itself)

Self-attention: An example

Sentence: *"The bank on Wall Street raised rates."*

When processing "bank":

- High attention to "Wall Street" $\rightarrow$ financial institution
- High attention to "raised rates" $\rightarrow$ confirms financial meaning

Transformer architecture: Key components

1. **Input embeddings**: Convert words to vectors (like Word2Vec)
2. **Positional encoding**: Add information about word order
3. **Multi-head self-attention**: Look at other words from multiple "perspectives"
4. **Feed-forward layers**: Process each position independently
5. **Layer normalization & residual connections**: Help training
6. **Stack many layers**: 12-24 layers for BERT, 96+ for GPT-4

Transformer architecture: Output and advantages

Output: Contextualized representation for each word

- Unlike Word2Vec, representation depends on surrounding words

Key advantage: Entire sequence processed in parallel $\rightarrow$ much faster training

Two flavors of transformers

BERT family:

- Reads entire text at once (bidirectional)
- Good for: Classification, NER, question answering
- Example: Sentiment analysis

GPT family:

- Generates text left-to-right (autoregressive)
- Can only look at previous words, not future words
- Good for: Text generation, completion
- Example: "Complete this sentence: The Federal Reserve..."

(Somewhat historical groupings)

BERT: Pre-training approach

Key idea: Pre-train on massive unlabeled text, then fine-tune for specific tasks

Pre-training objectives:

- **Masked Language Modeling (MLM)**:
  - Hide 15% of words, predict them from context
  - Example: “The Federal [MASK] raised rates” → predict “Reserve”
- **Next Sentence Prediction**:
  - Predict if sentence B follows sentence A
  - Helps understand relationships between sentences

BERT: Impact

Result: Rich contextual representations that work well for many tasks

Why it matters: Revolutionized NLP in 2018

- Showed power of pre-training + fine-tuning

ModernBERT (2024)

Recent improvements to BERT architecture:

- **Longer context**: 8,192 tokens (vs. 512 for original BERT)
- **Better efficiency**: Faster training and inference
- **Updated pre-training**:
  - Trained on more recent data
  - Better optimization techniques
  - No Next Sentence Prediction (didn't help much)
- **Strong performance**: Matches or beats larger models on many tasks

Key insight: Architecture improvements matter

- Not just about scale (bigger models), also engineering + training techniques

For researchers: ModernBERT is a good default choice for supervised learning

The rise of Large Language Models (LLMs)

Scaling up decoder-only transformers:

- GPT-3 (2020): 175B parameters
- GPT-4 (2023): $>$1T parameters (estimated)
- Claude, Gemini, Llama: Similar scale

Emergence of new capabilities at scale:

- **In-context learning**: Learn from examples in the prompt
- **Reasoning**: Chain-of-thought, step-by-step problem solving
- **Instruction following**: Do what you ask without fine-tuning
- **Multi-task**: One model, many tasks

LLM training paradigm

1. **Pre-training**: Predict next word on massive text corpus
2. **Instruction tuning**: Fine-tune on instruction-following examples
3. **RLHF**: Align with human preferences

Reinforcement Learning from Human Feedback (RLHF)

Problem: Pre-trained LLMs are good at predicting text, but not at being helpful

RLHF process:

1. **Collect human preferences**:
   - Show humans multiple model outputs for same prompt
   - Ask: "Which response is better?"
2. **Train reward model**:
   - Learn to predict human preferences
   - Input: (prompt, response) $\rightarrow$ Output: quality score
3. **Optimize policy**:
   - Use reinforcement learning to maximize reward
   - Make model generate responses humans prefer

RLHF: Results

Result: Models that are helpful, harmless, and honest

- Follow instructions better
- Refuse harmful requests
- Admit uncertainty

Reasoning capabilities

Chain-of-Thought (CoT) prompting:

- Ask model to "think step by step"
- Dramatically improves performance on reasoning tasks

Example:

Without CoT: "The Federal Reserve raised rates 4 times in 2022 and 3 times in 2023. How many total increases?" $\rightarrow$ Often incorrect

With CoT: "...Think step by step."
Model: "Let me break this down: 2022: 4 rate increases; 2023: 3 rate increases; Total: $4 + 3 = 7$ increases"

Why it works: Generating intermediate steps helps model reason