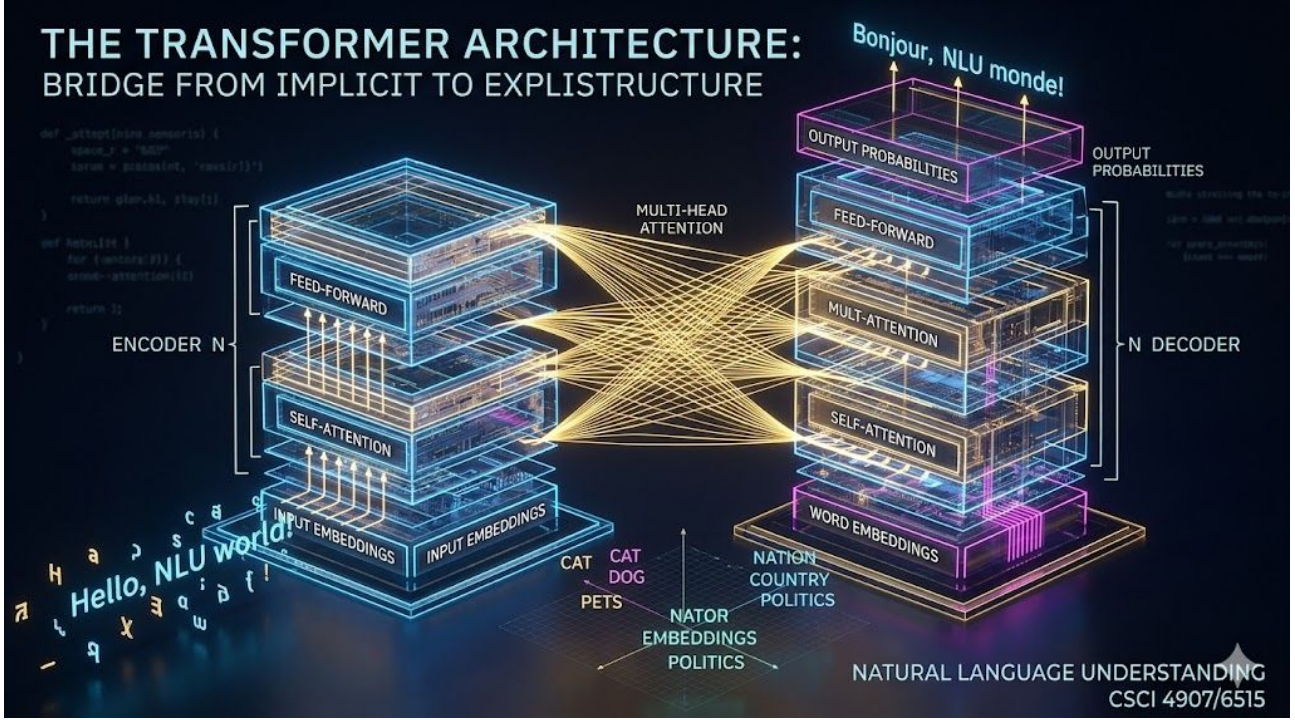


Natural Language Understanding
Deep Neural Networks & Transformer Models

CSCI 4907/6515

Lecture 10, March 24, 2026

Aya Zirikly

Stacked RNNs

We can't use standard RNNs for long documents. We need a new type of neuron: one that has a dedicated "Save" button and a "Delete" button so the math doesn't dilute over time.

Long Short-Term Memory

Long Short-Term Memory (LSTM)

- An LSTM is a highly upgraded, special type of RNN hidden layer.
- Instead of just passing one simple "short-term memory" forward, an LSTM creates two separate memory tracks: a **Short-Term Memory** (for immediate context) and a **Long-Term Memory** (a permanent track that runs straight through the entire paragraph).
- LSTMs can learn *when* to remember something and *when* to forget it. If it reads that the main character's name is "Alice" in chapter 1, it can, in theory, hold onto that specific fact all the way to chapter 10!

LSTM -How it works

The Conveyor Belt and the Three Gates

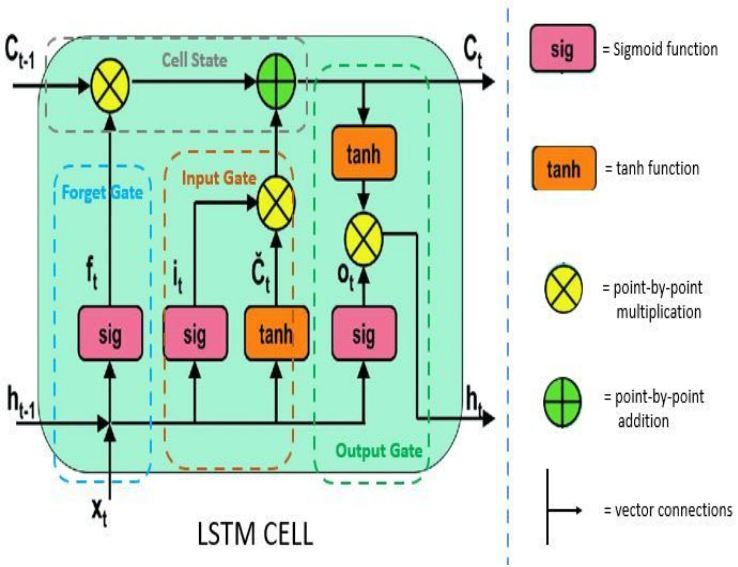
The Core: The Cell State (The Conveyor Belt)

- This is the long-term memory track. It acts like a high-speed conveyor belt running straight through the network. Information can easily flow down this belt unchanged unless a gate explicitly stops it.

LSTM -How it works

The Three Smart Gates:

1. **The Forget Gate (The Eraser):** It looks at the new word and the short-term memory, then looks at the conveyor belt. It decides what old information is now totally useless and deletes it. *(e.g., The sentence ended with a period, so erase the old subject).*
2. **The Input Gate (The Pen):** It decides which pieces of the new words are important enough to write onto the conveyor belt for permanent storage. *(e.g., The new sentence introduced a dog named Buster. Write "Buster" down).*
3. **The Output Gate (The Filter):** It looks at the massive amount of information on the conveyor belt and decides what specific piece is actually needed *right now* to guess the very next word.



LSTM working example

Let's watch an LSTM Language Model predict the final word of this sentence: *"Alice grew up in **France**. After moving to the US, she still speaks fluent _____."*

1. The Goal

- The correct prediction is **"French"**.
- But notice the massive gap! The critical clue ("France") is 10 words away from the blank space.
- A standard Recurrent Neural Network (RNN) would have forgotten "France" by the time it reached the word "fluent," and might just guess random languages like "Spanish" or "English."

2. The LSTM's Memory Track (The Cell State)

- When the LSTM reads the word "France" at the beginning of the sentence, its **Input Gate** recognizes this as an important noun and writes the concept of *[Location: France]* onto its long-term memory conveyor belt.
- As the network reads the filler words in the middle ("After, moving, to, the, US"), the memory conveyor belt protects that *[Location: France]* tag and carries it safely forward.

LSTM working example

Making the final prediction

"she still speaks fluent _____." How do the gates work together to predict the word "French"?

1. The Forget Gate (Clearing Clutter)

- The network just read the word "US" a few steps ago. The Forget Gate looks at the context and decides: *"She moved AWAY from the US, so we don't need this for her native language."* It weakens the "US" signal.

2. The Input Gate (Adding New Context)

- The network reads the word "speaks" and "fluent." The Input Gate writes a new rule onto the conveyor belt: *"We are now looking for a language."*

LSTM working example

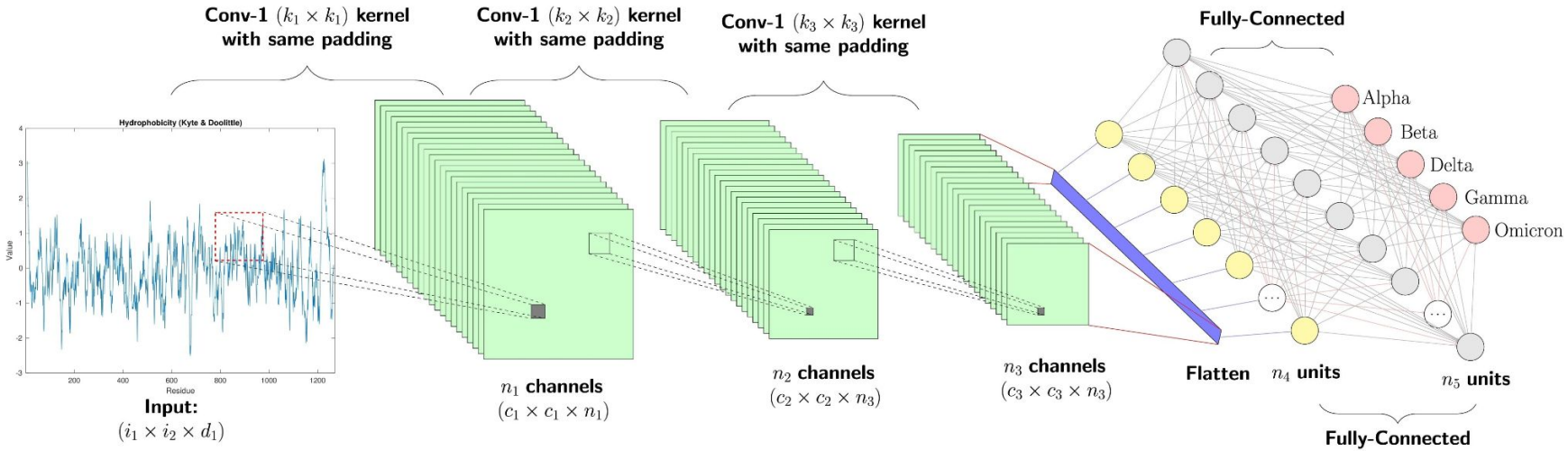
Making the final prediction

Now, the network reaches the end of the sentence: *"she still speaks fluent _____."* How do the gates work together to predict the word "French"?

3. The Output Gate (The Aha! Moment)

- The Output Gate acts as the final filter. It looks at the current request (*"We need a language"*) and scans the long-term memory conveyor belt (*"Ah, we still have [Location: France] saved from earlier!"*).
- It combines those two pieces of information, runs the math, and confidently outputs the highest probability for the word **"French"**!

Convolutional Neural Networks (CNN)



If RNNs and LSTMs are designed to process data over *Time* (word by word)

CNNs are designed to process data over *Space* (grids and localized regions)

CNN

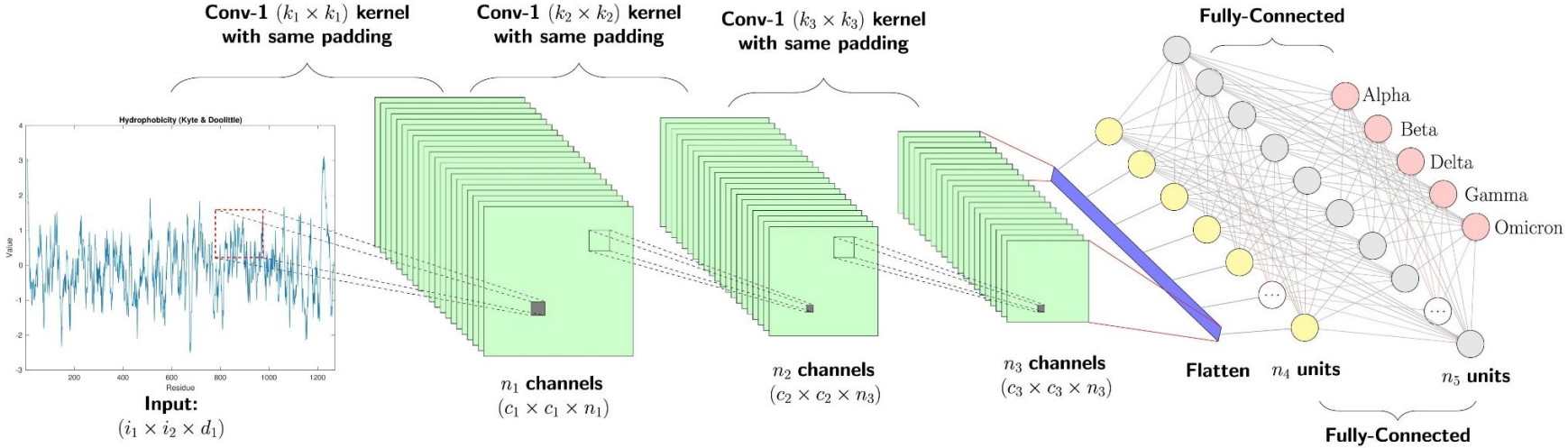
The core concept

- A standard Feed-Forward network looks at the entire input all at once. An RNN reads it one step at a time.
- A CNN looks at data through a **sliding window**. It scans small, localized chunks of data to find specific patterns, regardless of where they appear.

Where Do CNNs Work Best?

- **Computer Vision (2D Data):** excellent for image recognition, self-driving cars, and facial recognition. They scan grids of pixels to find edges, then shapes, then full objects.
- **Audio (2D Spectrograms):** Turning soundwaves into visual heatmaps allows CNNs to "look" at audio to recognize speech or music genres.
- **Natural Language Processing (1D Data):** By treating a sentence as a 1D line of words, a CNN can scan across the text to instantly spot strong phrases (like "highly recommend" or "terrible service") without having to read the entire sentence sequentially!

Convolutional Neural Networks (CNN)



CNN Architecture

A CNN works by passing the data through two highly specialized types of layers before making a final decision.

The Convolutional Layer (The Flashlight)

- Imagine shining a small flashlight (called a **Filter** or **Kernel**) over the top-left corner of an image, and then slowly sliding it across the whole picture.
- The Filter/Kernel is looking for one specific mathematical pattern (like a vertical line, or in NLP, a specific 2-word phrase). Every time it finds a match, it "lights up" and creates a **Feature Map**.

CNN Architecture

The Max Pooling Layer (The Summarizer)

- After the Filter creates a map of where it found patterns, the Pooling layer shrinks that map down.
- Look at a small block of the map and grab the maximum number (the strongest signal), throwing away the rest. This drastically reduces the memory required and makes the network focus only on the most important features!

The Fully Connected Layer (The Judge)

- All those summarized maps are flattened out and handed to a standard Dense Neural Network to make the final prediction (e.g., "This is a picture of a cat" or "This is a Positive review").

Working Example

Let's watch a 1D CNN classify the sentiment of this movie review: *"The movie was completely boring and too long."*

Step 1. *The sentence is converted into word embeddings, forming a 1D line of math.*

Step 2. **The Sliding Filter (Size = 2 words)**

The network's Filter is trained to look for negative 2-word phrases (bigrams). It slides across the sentence like a magnifying glass:

1. [*The movie*] → No strong signal.
2. [*movie was*] → No strong signal.
3. [*completely boring*] → **Massive negative signal spike!**
4. [*too long*] → **Another strong negative signal spike!**

Working Example

Let's watch a 1D CNN classify the sentiment of this movie review: ***"The movie was completely boring and too long."***

Step3. Max Pooling

- The Pooling layer looks at the entire signal map for the sentence. It sees a bunch of zeros, but two massive spikes. It grabs the strongest spikes (*"completely boring"*) and discards the boring filler words.

Step 4. The Output layer looks at the pooled evidence. It doesn't care *where* the phrase "completely boring" was in the sentence. It just knows the phrase was detected loud and clear. It confidently predicts: **Negative Sentiment (0)**.

Summary

Feature	CNNs (Convolutional)	RNNs & LSTMs (Recurrent)
How It Reads	**Spatial:** Scans for local clumps of words (n-grams) simultaneously.	**Sequential:** Reads one word at a time, strictly left-to-right.
Speed	**Extremely Fast:** Can process the whole sentence in parallel at the exact same time.	**Slower:** Has to wait for word 1 to finish before it can process word 2.
Core Strength	**Keyword Spotting:** Finding strong signals regardless of where they are in the text.	**Context & Order:** Understanding grammar, long distances, and the flow of time.
Best NLP Tasks	Sentiment Analysis, Spam Detection, Topic Categorization.	Machine Translation, Text Generation, Question Answering.

NLP Tasks

… and how they used these techniques

Named Entity Recognition (NER)

NER is an information extraction task. The goal is to locate and classify named entities in unstructured text into predefined categories.

What exactly counts as a "Named Entity"?

Simply put, it is a real-world object, person, or concept that has a specific, unique, and proper name.

NER

Standard

PER (Person): *Nelson Mandela, Dr. Smith*

ORG (Organization): *Google, The United Nations, George Washington University*

LOC (Location): *Tokyo, the Sahara Desert, Mars*

MISC (Miscellaneous): *The iPhone (Product), The Mona Lisa (Work of Art), Spanish (Language/Nationality)*

Numerical

- **DATE / TIME:** *October 31st, 1999, Next Tuesday, 4:00 PM*
- **MONEY / QUANTITY:** *-->5 million, 500 kilograms, 20%*

NER: *The Ambiguity Problem*

Why can't we just give a computer a massive dictionary of names and companies and tell it to find matches? Because in human language, **context** is everything. A single word can completely change its identity depending on the words next to it!

The "Chameleon" Words Look at how the exact same string of letters requires completely different NER tags based entirely on context:

- **The "Apple" Problem:**
  - *"I ate a delicious **apple** for lunch."* → **O** (Generic Noun / Fruit)
  - *"**Apple**'s stock went up today."* → **B-ORG** (Organization)
- **The "Washington" Problem:**
  - *"**Washington** crossed the Delaware River."* → **B-PER** (Person)
  - *"She booked a flight to **Washington**."* → **B-LOC** (Location)
  - *"The **Washington** Post published an article."* → **B-ORG** (Organization)

NER: *The Ambiguity Problem*

Why Traditional Systems Failed If you build a simple rule-based system that says *"Always tag 'Apple' as a Company,"* your AI will start telling your users that they are eating an Organization for lunch!

The Need for "Deep Context" To solve this, an AI cannot just read the target word. It has to read the *entire sentence* to pick up on clues. It needs to learn that words like "ate" and "delicious" usually appear near fruits, while words like "stock" and "CEO" appear near companies.

NER: example

- **Raw Text:** *"Tim Cook announced that Apple will open a new campus in Austin next September."*
- **NER Output:** * [**Tim Cook**] → PERSON
    - [**Apple**] → ORGANIZATION
    - [**Austin**] → LOCATION
    - [**next September**] → DATE / TIME

3. Why Do We Care?

- **Customer Support:** Automatically routing support tickets based on the product mentioned.
- **Search Engines:** Identifying what a user is actually searching for.
- **Finance:** Scanning thousands of news articles to extract company names and locations for trading algorithms.

How the machine reads it: *The BIO Tagging System*

If we are treating NER as a "Sequence Labeling" task (where every single word gets a label), how do we handle multi-word entities like "New York City"? We use the **BIO Format** (Begin, Inside, Outside).

The BIO Rules:

- **B (Begin):** The first word of an entity.
- **I (Inside):** Any subsequent words in that same entity.
- **O (Outside):** Filler words that are not entities.

Example applied to a sentence:

- Tim → **B-PER** (Begin Person)
- Cook → **I-PER** (Inside Person)
- visited → **O** (Outside)
- New → **B-LOC** (Begin Location)
- York → **I-LOC** (Inside Location)

Evaluation

Precision (Quality)

- *Out of all the entities the AI guessed, how many were actually right?*
- If it highlights 10 words as "Locations," but only 7 are real locations, the Precision is 70%.

Recall (Quantity)

- *Out of all the true entities hidden in the text, how many did the AI manage to find?*
- If there are 10 real locations in the text, and the AI only finds 4 of them, the Recall is 40%.

F1-Score

- The F1-Score is the harmonic mean of Precision and Recall. It forces the model to be good at *both* finding everything (Recall) and not making false guesses (Precision).
- **Formula:** F1 = 2 * (Precision * Recall) / (Precision + Recall)

Pre-neural NER: *Feature Engineering and CRFs*

Rule-Based Systems (Regex)

- Programmers wrote massive dictionaries of names and complex "Regular Expressions" (rules like: *If a capitalized word comes after "Mr.", it is a Person*).
- **The Flaw:** Language is too messy. You can't write a rule for every exception!

Traditional Machine Learning (CRFs)

- We moved to statistical models like **Conditional Random Fields (CRFs)**.
- To make them work, humans had to do "Feature Engineering." We had to manually tell the AI what clues to look for: *"Is the word capitalized? Is it a noun? What is the word next to it?"*
- **The Flaw:** Human engineers had to manually invent every single clue.

Entering neural net era: *let the model find its own clues*

Word Embeddings

- Instead of looking at capitalization rules, words were converted into dense math vectors (like Word2Vec or GloVe) that captured their actual semantic meaning.
  - Because "London" and "Tokyo" are used in similar contexts, their mathematical coordinates are placed right next to each other in this space.
- **RNNs and LSTMs Took Over**
- Since NER requires reading a sequence of words, the **LSTM** became the go-to architecture.
- It could read a sentence left-to-right, remember the context, and output a BIO tag for each word.
- **The Flaw:** If you only read left-to-right, you might miss clues at the end of the sentence that explain the beginning! (e.g., *"Apple is a great... fruit vs. company"*)

Pre-BERT: *The BiLSTM-CRF Architecture*

Right before Transformers and BERT changed the world in 2018, the NLP community created **The BiLSTM-CRF**: successful and very popular

Conditional Random Fields (CRF)

- **Motivation**. Independent guessing: If a basic neural network predicts entities word-by-word independently, it makes illogical mistakes. It might tag the word "New" as a Location, but tag the following word "York" as a Person simply because "York" is a common last name. It lacks structural awareness.
- A Conditional Random Field (CRF) is a probabilistic mathematical layer added to the very end of a neural network. Instead of just looking at the input words, a CRF evaluates the **transitions between the predicted tags** to ensure the final sequence obeys logical rules.

Conditional Random Fields (CRF)

The BIO Format Example In Named Entity Recognition, we use the BIO tagging scheme (Begin, Inside, Outside).

- "New York" should be tagged as **B-LOC** (Begin Location) and **I-LOC** (Inside Location).
- A CRF learns structural rules from the training data: *An **I-LOC** tag can never appear unless a **B-LOC** tag comes immediately before it.* * If the underlying neural network mistakenly guesses an **O** (Outside) followed by an **I-LOC**, the CRF steps in, recognizes that this transition is mathematically impossible, and corrects the sequence!

Calculating the Best Path: Emissions and Transitions

To find the most logical sequence of tags for a sentence, the CRF calculates a final "Score" by adding together two distinct values for every single word:

The Emission Score (The Word's Appearance) This score comes from the underlying neural network (like a BiLSTM). It measures how much the word *can be a* specific entity, regardless of context.

- *Example:* The word "York" has a high Emission Score for the `Location` tag.

The Transition Score (The Sequence Logic)

This score comes directly from the CRF layer. It measures how likely Tag A is followed by Tag B based on the rules of human language.

- *Example:* The transition from B-LOC to I-LOC has a very high Transition Score. The transition from B-PER (Person) to I-LOC has a Transition Score of near zero.

Calculating the Best Path: Emissions and Transitions

To score a potential path (Y) for a sentence (X), the CRF simply sums up the Emissions and Transitions:

$$\text{Score}(X, Y) = \Sigma \text{ Emission}(x_i, y_i) + \Sigma \text{ Transition}(y_{i-1}, y_i)$$

Finding the Winner The CRF calculates this score for all possible tag combinations. It then converts those scores into probabilities (using a Softmax function) and selects the single sequence of tags with the highest overall probability, guaranteeing a grammatically flawless output.

Pre-BERT: *The BiLSTM-CRF Architecture*

Right before Transformers and BERT changed the world in 2018, the NLP community created **The BiLSTM-CRF**: successful and very popular

1. The BiLSTM (Bidirectional LSTM) for Context

- It didn't just use one LSTM; it used two! One read the sentence left-to-right, and the other read it right-to-left simultaneously.
- This allowed the network to know exactly what came *before* and *after* a word before guessing its tag.

Pre-BERT: *The BiLSTM-CRF Architecture*

2. The CRF Layer for Rules

- Even with a BiLSTM, the neural network sometimes impossible guesses (like outputting an **I-LOC** without a **B-LOC** first).

- Engineers slapped a traditional **CRF** layer on top of the BiLSTM. The CRF acted as a strict grammar police officer, enforcing the BIO sequence rules and making sure the final tags made logical sense together.

NER in the Hospital (Physical Health)

Reading the Doctor's Messy Notes

Hospitals have mountains of unstructured Electronic Health Records (EHRs): typed notes from doctors that databases couldn't easily search. NER was used to turn those messy paragraphs into clean data.

The Medical Entities (The "Buckets") Instead of Person, Organization, and Location, clinical NER uses entirely different tags:

- **Medical Condition / Disease:** *Type 2 Diabetes, Hypertension, Asthma*
- **Sign / Symptom:** *Nausea, chest pain, fever*
- **Medication / Drug:** *Lisinopril, Ibuprofen*
- **Dosage & Route:** *50mg, orally, twice daily*
- **Procedure:** *Appendectomy, MRI scan*

Reading the Doctor's Messy Notes

The Ambiguity Problem in Medicine Pre-BERT models (like CRFs) struggled heavily here because medical text is full of abbreviations and double meanings.

- Does **"pt"** mean *Patient* or *Physical Therapy*?
- Does **"discharge"** mean *fluid leaving the body* or *the patient leaving the hospital*?

NER in Mental Health

Extracting entities for mental health was, and still is, vastly more difficult than physical health. Why? Because mental health is highly descriptive, subjective, and rarely uses strict medical terminology.

What are the entities? Researchers scanning clinical notes or social media (like Reddit or Twitter) for mental health studies tried to extract:

- **Psychiatric Symptoms:** *Insomnia, anhedonia (loss of interest), panic attacks*
- **Stressors / Triggers:** *Divorce, job loss, bankruptcy*
- **Substance Use:** *Alcohol, opioids, withdrawal*

NER in Mental Health

Why Pre-BERT Models Struggled

- **Physical Health is Explicit:** *"Patient has a fractured femur."* (Easy for a BiLSTM or Dictionary to spot).
- **Mental Health is Implicit:** *"I just don't see the point in getting out of bed anymore."*

  To an older NER system, there is no explicit "disease" word in that sentence! Older models required strict feature engineering or dictionaries, which failed to capture the nuances of human emotion and depression.

Clinical NER in 2010s

Before Transformers, the healthcare industry relied heavily on massive dictionaries combined with early machine learning.

The Dictionary Giants (cTAKES & MetaMap)

- Hospitals used software that linked directly to the **UMLS (Unified Medical Language System)**: a massive government database of every medical term ever recorded.
- These tools used strict rule-based matching. If the doctor typed "heart attack," the software explicitly mapped it to the official entity "Myocardial Infarction."

Clinical NER in 2010s

The Transition to Machine Learning (i2b2/n2c2 Challenges)

- As dictionaries failed to understand context, medical researchers hosted famous competitions (like the i2b2/n2c2 NLP challenges).
- By 2017, the absolute best systems for medical NER were **BiLSTM-CRFs**. They could read a sentence like *"Patient denies experiencing chest pain,"* and the BiLSTM would successfully link the word "denies" to "chest pain" so the system wouldn't accidentally tag the patient as currently having a heart attack!

The Transformer and BERT

The problem with reading left-to-right

Before 2017, the BiLSTM was the king of AI. But it had a fatal flaw: **It was sequential.**

- **The Speed Limit:** It had to read word n, then word n+1, then word n+2. You could not process a sentence in parallel. It was incredibly slow to train.
- **The Fading Memory:** Remember our "Alice in France" example? Even with a conveyor belt, if the paragraph is 500 words long, the LSTM's memory still degrades over time.
- **The Goal:** How do we build a network that reads the *entire* paragraph all at the exact same time, just like a CNN, but still understands the grammar and order?

Attention Is All You Need

In 2017, Google researchers published a paper that threw away RNNs and LSTMs entirely. They introduced **The Transformer**.

- Instead of passing a memory state left-to-right, the Transformer uses an "Attention Mechanism."
- **Attention:** a mathematical system that allows the AI to look at every single word in a sentence simultaneously and calculate exactly how much "attention" each word should pay to every other word.

Attention is all you need. Vaswani et al. 2017

The math of Self Attention

To calculate exactly how much context a word should absorb from its neighbors, the Transformer runs each word through a strict mathematical sequence using three distinct vectors.

The Q, K, V Projections

Every word in the sentence (represented as an embedding vector x) is multiplied by three separate learned weight matrices (W^Q, W^K, W^V) to create three new vectors:

- **The Query (Q):** What this word is looking for (e.g., *"I am an adjective looking for a noun"*).
- **The Key (K):** What this word contains (e.g., *"I am a plural noun"*).
- **The Value (V):** The actual underlying semantic meaning of the word.

The math of Self Attention

Calculating Similarity (The Dot Product)

To figure out how much the word "apple" should pay attention to the word "delicious", the network calculates the **dot product** between the Query of "apple" and the Key of "delicious".

- $Q \cdot K^T$ calculates the geometric alignment between the two vectors. A higher dot product means the words are highly relevant to each other.

The math behind Self-Attention

Scaling and Softmax (Normalization)

- The raw dot products can get extremely large, which ruins the neural network's gradients.
  - Divide by $\sqrt{d_k}$
- The network applies a standard **softmax** function. This converts the raw scores into a clean probability distribution (percentages that sum up to 1).

The Contextual Update (The Weighted Sum)

Finally, the network multiplies these softmax percentages by the **Value (V)** vectors of the surrounding words and adds them all together. This creates a brand new, context-heavy vector for the target word!

$$\text{Attention}(Q, K, V) = \text{softmax}\left((Q \times K^T) / \sqrt{d_k} \right) \times V$$

Single head vs. Multi-head attention

A single attention head runs the `Attention(Q, K, V)` math exactly one time.

- Because it only calculates one set of scores, it tends to focus heavily on just *one* type of relationship.
- *Example:* For the sentence *"The tired doctor quickly wrote the prescription"*, a single head might only learn to connect the adjective ("tired") to the noun ("doctor"). But it might completely miss the connection between the adverb ("quickly") and the verb ("wrote").

Single head vs. Multi-head attention

the Transformer splits the input and runs the exact same attention math **multiple times in parallel** using different learned weights for each run. Each parallel run is called a "Head."

- Standard BERT uses **12 distinct Heads** at the exact same time!

Different Heads Learn Different Things Because the network is training 12 different heads simultaneously, they naturally divide up the labor.

- **Head 1:** Might become the "Grammar Expert" (focusing strictly on subject-verb pairs).
- **Head 2:** Might become the "Pronoun Expert" (focusing entirely on connecting "he/she/it" to the right person).
- **Head 3:** Might become the "Emotion Expert" (linking negative words together).

INSIDE THE TRANSFORMER — ATTENTION MECHANISM

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^T / \sqrt{d_k} \right) \cdot V$$

Q

Query

What am I looking for?

K

Key

What do I contain?

V

Value

What info do I give?

$\sqrt{d_k}$

Scale

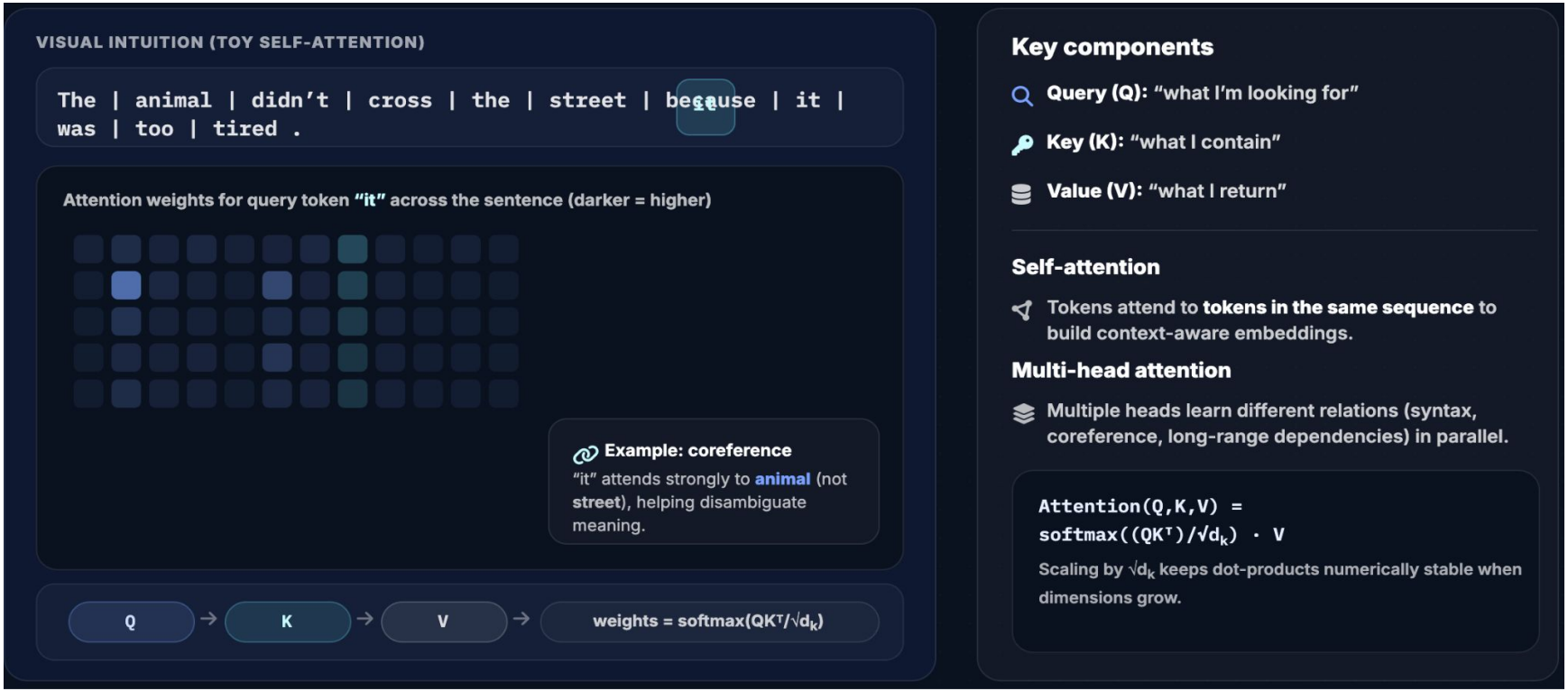
Prevents softmax saturation

Causal Masking — What Each Token Can 'See'

	The	cat	sat	on	mat
The	✓	✗	✗	✗	✗
cat	✓	✓	✗	✗	✗
sat	✓	✓	✓	✗	✗
on	✓	✓	✓	✓	✗
mat	✓	✓	✓	✓	✓

✓ teal = can attend (past) ✗ gray = masked (future)

Each token computes a weighted average over other tokens—so the model can “focus” on what matters for the current context.



Fixing the ambiguity problem

Ambiguity with: Apple?

Sentence: *"I ate a delicious green apple."*

When the Transformer looks at the word "apple," it doesn't just see the word. It casts a net over the whole sentence.

The attention mechanism mathematically links "apple" heavily to the words "ate" and "delicious."

It dynamically updates the meaning of the word "apple" based on its neighbors *before* it makes any predictions! It literally bakes the context directly into the word.

BERT: *Bidirectional encoder representations from transformers*

A year after the Transformer was invented, researchers at Google released **BERT** (Bidirectional Encoder Representations from Transformers).

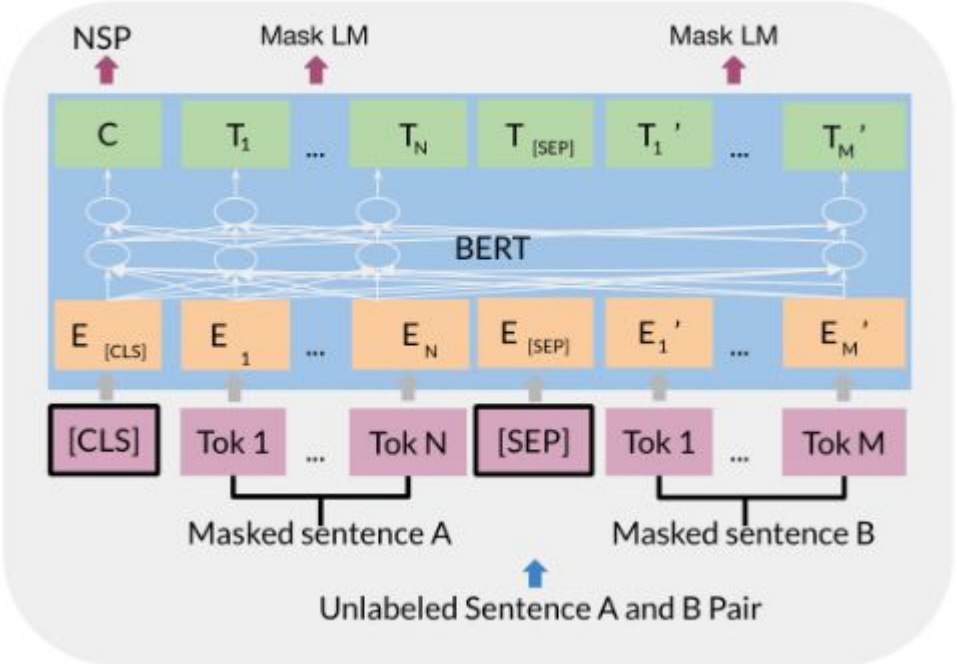
- **The "BiLSTM" on Steroids:** The BiLSTM read left-to-right, and then right-to-left. BERT doesn't do that. Because of Self-Attention, BERT reads the entire sentence in all directions simultaneously.
- **Deep Context:** It knows exactly what is to the left of a word and to the right of a word at the exact same time, giving it the deepest understanding of language context ever achieved.

Reading in Both Directions at Once

BERT architecture

CLS: classification token (summary)

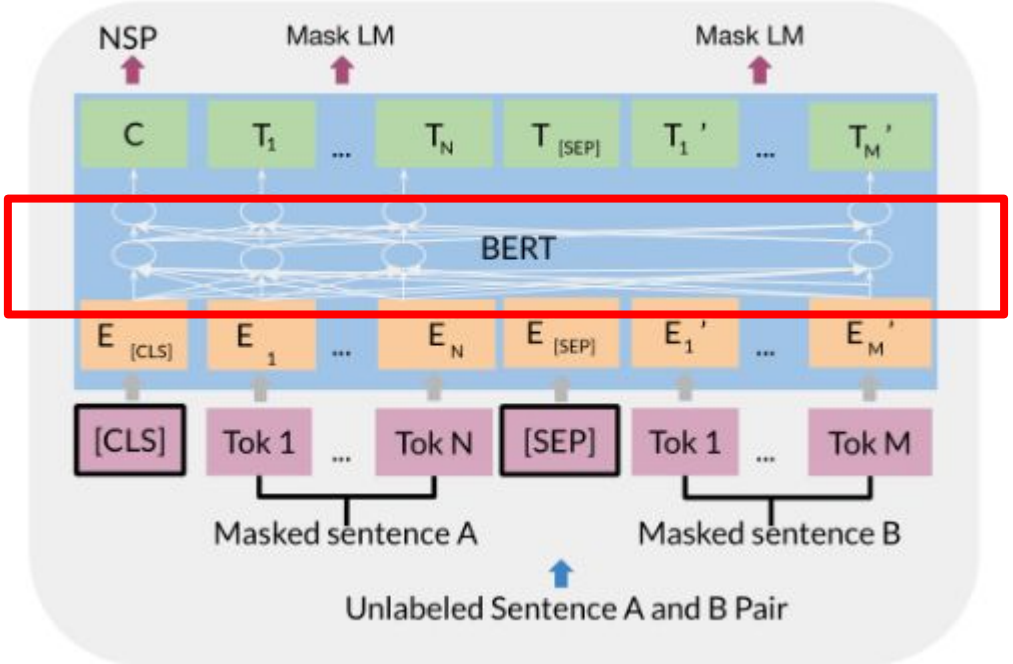
SEP: separator between sentences



BERT architecture

CLS: classification token (summary)

SEP: separator between sentences



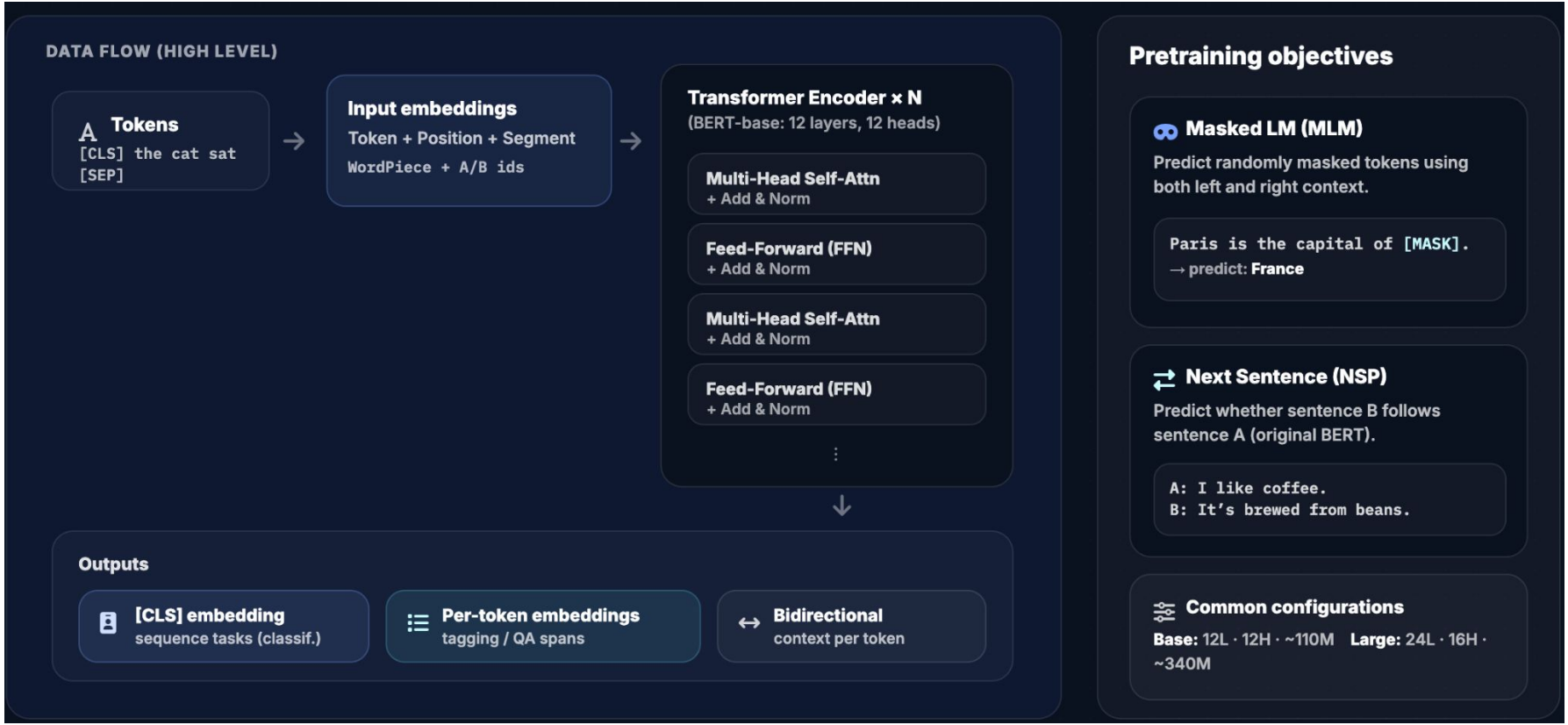
How BERT Learned to Read: *Masked Language Modeling*

The Process: take entire English Wikipedia, randomly masked 15% of the words, and asked BERT to guess the missing words.

Example: *"The man went to the [MASK] to buy some milk."*

By forcing the network to guess millions of missing words, BERT was forced to learn English grammar, facts, and deep context entirely on its own!

BERT Architecture



Key takeaways

3 big ideas

Attention = weighted token mixing

Each token builds context by attending to others. Multi-head attention learns different relations in parallel (e.g., syntax, coreference, long-range links).

$$\text{softmax}((QK^T)/\sqrt{d_k}) \cdot V$$

BERT = encoder-only, bidirectional context

Stacks Transformer encoders to produce contextual embeddings for every token. Pretraining with **MLM** enables transfer to many tasks.

```
Paris is the capital of [MASK]. → France
```

Outputs map cleanly to tasks

Use **[CLS]** for sequence classification. Use **per-token embeddings** for tagging and QA (start/end span).

`[CLS] → label` `tokens → span/tag`

Practical checklist (fine-tuning)

- Always pass an **attention mask** for padding (prevents attending to pad tokens).
- Mind the **max sequence length** (BERT-base typically 512 tokens): truncate, chunk, or use long-context models.
- **Tokenization matters:** WordPiece splits rare words; keep preprocessing consistent between training and inference.
- **Choose the right model & metrics**
  Use DistilBERT/ALBERT for speed; Base/Large for accuracy. Track Accuracy/F1 (classification) and EM/F1 (QA). Watch domain drift.

Fine tuning revolution

Before BERT

- ❏ if you wanted an NER model, you built one from scratch
- ❏ If you wanted a translation model, you built a new one from scratch.

After BERT

- BERT is a pre-trained "base brain." It already knows how to read English perfectly.
- **Fine-Tuning:** To make an NER system, you just take the BERT model, show it a few hundred examples of Named Entities and you have a trained BERT-based NER model.

The GLUE Benchmark

The General Language Understanding Evaluation (GLUE) benchmark is a standardized collection of diverse NLU tasks. It was developed to evaluate the generalization capabilities of machine learning models across a broad spectrum of linguistic challenges, rather than assessing performance on a single, isolated dataset.

- **Single-Sentence Tasks:** Evaluates fundamental linguistic acceptability and sentiment (e.g., Corpus of Linguistic Acceptability [CoLA], Stanford Sentiment Treebank [SST-2]).
- **Similarity and Paraphrase Tasks:** Measures the ability to determine semantic equivalence between sentence pairs (e.g., Microsoft Research Paraphrase Corpus [MRPC], Quora Question Pairs [QQP]).
- **Inference Tasks:** Assesses logical deduction and entailment, determining if a premise mathematically or logically supports a hypothesis (e.g., Multi-Genre Natural Language Inference [MNLI], Recognizing Textual Entailment [RTE]).

The GLUE Benchmark

- Single, comprehensive leaderboard and scoring metric
- Unified evaluation metric
- GLUE catalyzed rapid advancements in the field, serving as the primary proving ground for paradigm-shifting architectures such as BERT and GPT.

How much better was BERT than the old LSTM and CRF models?

Pre-BERT models scored around 70 percent (macro-avg). BERT immediately jumped the score to over 80 percent, effectively beating human baselines on several reading comprehension tests for the first time in history.

Coreference Resolution

Older models struggled when sentences got complex: *"The trophy didn't fit into the brown suitcase because **it** was too large."*

What does "it" refer to? The trophy or the suitcase?

Because BERT's attention mechanism looks at the whole sentence, it easily links "it" to "trophy" (because trophies are large), solving a problem that plagued AI for decades.

Coreference Resolution

In the NLP community, this specific type of pronoun problem is called a **Winograd Schema**, and it is designed specifically to test if an AI has "commonsense" or "world knowledge."

The Need for World Knowledge

- Resolving ambiguous pronouns often requires an understanding of physics, spatial reasoning, or human behavior: concepts collectively known as "world knowledge."
- Traditional models failed these tasks because they only understood grammar, not reality.

Coreference Resolution

The Winograd Schema Challenge

Consider this classic linguistic test where changing a single adjective completely shifts the target of the pronoun:

- "*The trophy didn't fit into the brown suitcase because **it** was too **large**.*" (World Knowledge: The object being inserted is too big. Therefore, **it** = trophy).
- "*The trophy didn't fit into the brown suitcase because **it** was too **small**.*" (World Knowledge: The container is lacking space. Therefore, **it** = suitcase).

Coreference Resolution

BERT acquires *implicit* world knowledge during its **Masked Language Modeling (pre-training)** phase. Because:

- ❏ It was forced to read and predict missing words across the entire English Wikipedia and thousands of published books
- ❏ It mathematically encoded the statistical relationships between physical objects and their attributes (e.g., learning how "containers," "fitting," and "size" contextually interact in human language).

The leap performance in general NER

When applied to standard Named Entity Recognition (finding People, Orgs, Locations), BERT made traditional BiLSTM-CRF models obsolete.

- **Why it won?** An LSTM might get confused by the sentence *"Washington flew to Washington."*
- BERT’s attention mechanism instantly sees the word "flew" next to the first Washington (tagging it as a Person) and the word "to" next to the second Washington (tagging it as a Location). It achieved F1-scores of over 92 percent on standard NER datasets.

Examples

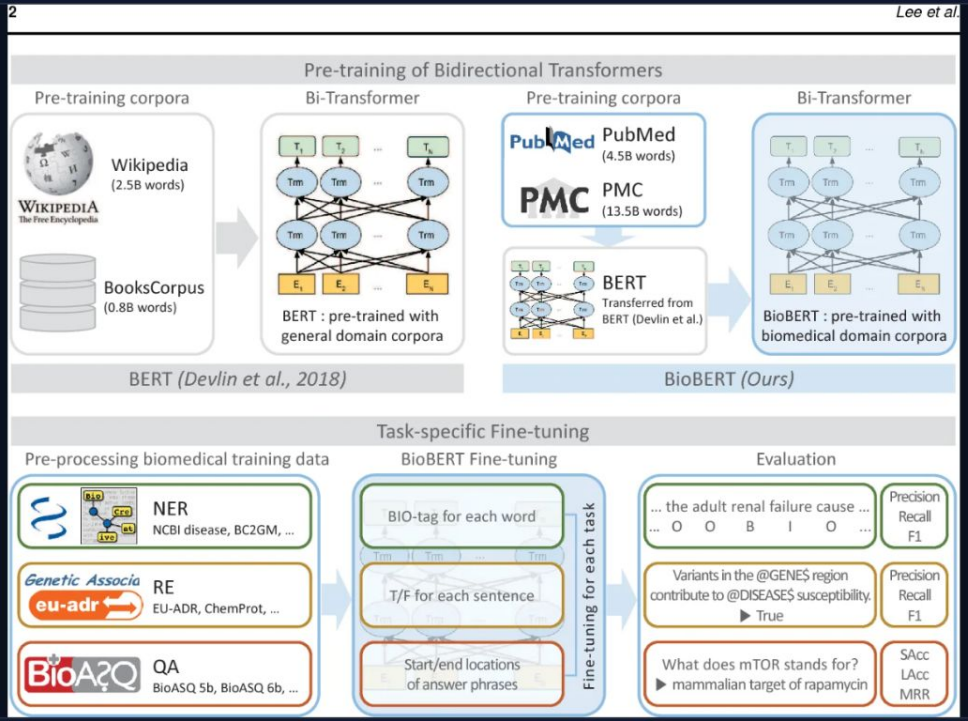
ClinicalBERT and BioBERT

- Standard BERT read Wikipedia.
- Models like **BioBERT** and **ClinicalBERT** were trained entirely on PubMed articles and millions of messy Electronic Health Records (EHRs).
- They learned the deep, complex "neighborhoods" of medical jargon that standard models couldn't understand.

BioBERT (The Foundation of Biomedical NLP)

- Developed by researchers in 2019
- BioBERT was one of the very first domain-specific adaptations of the original BERT model.
- Instead of just relying on standard Wikipedia English, the creators took the base BERT and continually trained it on billions of words from biomedical research articles in PubMed and PMC.
- It became the gold standard "base brain" for medical NER, excelling at identifying explicit diseases, drugs, and chemical compounds that standard BERT simply didn't understand.

FIGURE FROM BIOBERT PAPER (HIGH-LEVEL PRETRAINING & TASK ADAPTATION)



Source: Lee et al., "BioBERT: a pre-trained biomedical language representation model for biomedical text mining" (2019) • Image via Semantic Scholar

What is BioBERT?

BioBERT starts from BERT weights and continues pretraining on biomedical text (e.g., PubMed/PMC) using the same core objectives (notably **MLM**), improving representations for biomedical NLP.

WHY IT HELPS

- **Learns biomedical vocabulary & phrasing**
- **Better entity & relation context**
- **Strong transfer with minimal task heads**

Common applications

- Named Entity Recognition (genes, diseases, drugs)
- Relation extraction (drug–drug, protein–protein)
- Biomedical QA / document retrieval

ClinicalBERT (The Hospital Notes Specialist)

- ClinicalBERT (developed by researchers from MIT and Harvard) trained on the MIMIC-III database: a massive collection of over 2 million highly messy, unstructured Electronic Health Records (EHRs) and intensive care unit (ICU) doctor's notes.
- Because it learned the unique shorthand, abbreviations, it vastly outperforms standard models on hospital-specific NER tasks.

ClinicalBERT: BERT for Clinical Notes

BERT adapted to the clinical domain by **continued pretraining on EHR notes** (commonly MIMIC-III), improving understanding of clinical language and abbreviations.

VISUAL: CLINICALBERT & BLUEBERT OVERVIEW (ILLUSTRATIVE)

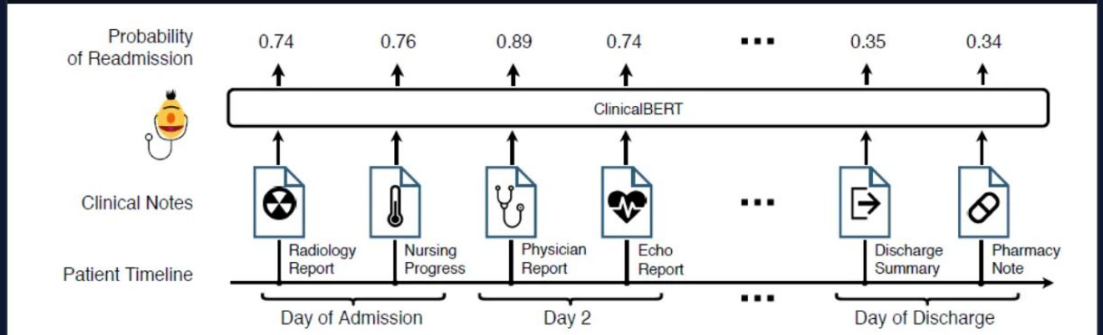


Figure 1: ClinicalBERT learns deep representations of clinical notes that are useful for tasks such as readmission prediction. In this example, care providers add notes to an electronic health record during a patient's admission, and the model dynamically updates the patient's risk of being readmitted within a 30-day window.

Source: "ClinicalBERT and BlueBERT" explainer graphic (Medium). For original ClinicalBERT reference: Huang et al., "ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission" (2019, arXiv:1904.05342).

What is ClinicalBERT?

ClinicalBERT initializes from BERT and continues pretraining on clinical text (typically **MIMIC-III** notes) using masked language modeling, producing embeddings better aligned with EHR language.

WHY IT HELPS

- Handles clinical jargon & abbreviations
- Stronger representations for note-based tasks
- Useful with small labeled clinical datasets

Common applications

- Clinical NER (problems, meds, labs)
- Phenotyping & cohort selection from notes
- Risk prediction (e.g., readmission) using note text

MentalBERT

- Mental health symptoms rarely use strict clinical jargon; patients describe their feelings in raw, highly conversational language.
- To solve this, researchers built MentalBERT (Ji et al., 2021): specifically fine-tuned on millions of mental health-related posts from Reddit (including specific support forums for depression, anxiety, and stress).
- Because it learned the *implicit* ways people express psychological distress online, it is currently one of the best architectures for extracting behavioral and mental health entities from social media and clinical therapy notes.

‘MentalBERT: BERT for Mental Healthcare’

BERT adapted to the mental health domain by continued pretraining on Reddit mental health forums, improving understanding of informal, psychological language and distress signals.

VISUAL: MENTALBERT OVERVIEW (ILLUSTRATIVE)

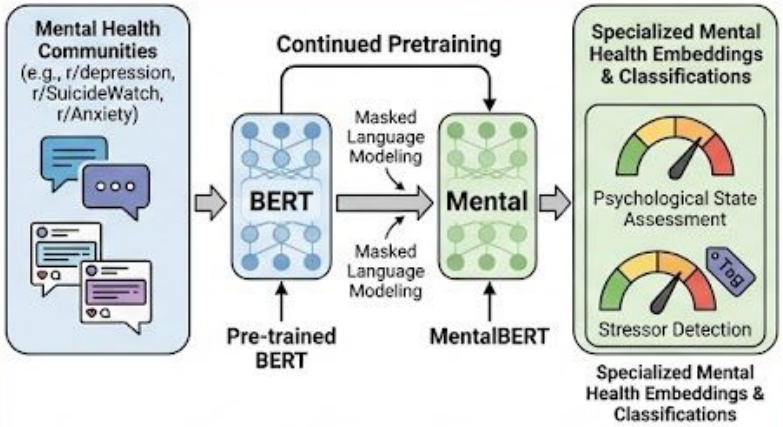


Figure 1: MentalBERT learns deep representations of informal, conversational text that are useful for mental health tasks. In this example, the model processes social media posts to dynamically assess a user’s psychological state or detect specific stressors.

What is MentalBERT?

Initializes from standard BERT and continues pretraining on massive, specialized corpora from mental health communities (typically Reddit forums) using masked language modeling. This produces produces embeddings deeply aligned with how people colloquially express psychological distress.

WHY IT HELPS

- **Understands implicit context:** Decodes conversational slang and metaphors that patients use instead of strict medical jargon.
- **Stronger representations for social/therapy text:** Better captures the subtle nuances of mood, stress, and ideation than standard models.
- **Bridges the patient-provider gap:** Translates raw patient feelings into recognizable clinical patterns.

Common applications

- Mental health NER (extracting stressors, behavioral symptoms, and triggers)
- Suicide ideation and severe risk detection
- Depression, anxiety, and stress classification from unstructured text

IMPROVED OUTCOMES

- Significantly more accurate identification of nuanced psychological distress in informal text.
- Enhanced ability to detect subtle behavioral cues and risk indicators.
- Improved classification accuracy across complex mental health conditions compared to standard models.

Source: Ji et al., "MentalBERT: Publicly Available Pretrained Language Models for Mental Healthcare" (2021, arXiv:2110.15621).

Solving Medical Ambiguity

Remember our older medical NER slides where the AI got confused by abbreviations?

The "Discharge" Problem:

- *"Patient is ready for **discharge** today."* (Attention links to "ready" → Tags as Event).
- *"Patient has purulent **discharge**."* (Attention links to "purulent" -> Tags as Symptom).

The "pt" Problem: *"The **pt** is experiencing pain."* (Attention links to "pain" → Tags as Patient).

- *"Scheduled for **pt** at 4 PM."* (Attention links to "scheduled" -> Tags as Physical Therapy).

Understanding the Implicit Context

Older models failed on: *"I just don't see the point in getting out of bed anymore."*

The BERT Leap: Because BERT was trained on billions of conversational phrases, its attention mechanism understands that the *entire combination* of those words equals the semantic concept of "Depressive Symptom."

It no longer relies on explicit dictionaries; it understands the human sentiment behind the text!

- What happens after we extract this sensitive data?
- How we do NER today?

De-identification and Data Privacy (HIPAA)

De-identification and Data Privacy (HIPAA)

- **Why do we need De-identification?** De-identification is essential because it strips away identifying personal details from medical records, allowing researchers and AI models to safely analyze massive amounts of valuable health data without violating patient privacy.
- **What is HIPAA?** HIPAA (the Health Insurance Portability and Accountability Act) is a strict federal law in the United States that establishes national standards to protect individuals' medical records and sensitive personal health information from being disclosed without their consent or knowledge.

De-identification

The "Reverse" NER Problem Usually, we use NER to extract data to *keep* it. In de-identification, we use NER to extract data to *destroy* it. We have to find every Patient Name, Doctor Name, Hospital Location, Date, and Phone Number, and mask them.

Why Dictionaries Fail Here

- If a hospital tries to scrub records by just deleting every word in a "Names Dictionary," it will accidentally delete the word "Hope" (a patient's name, but also a medical concept) or "Parkinson" (a patient's name vs. Parkinson's Disease).
- **The BERT Solution:** Because ClinicalBERT understands deep context, it knows exactly when "Hunter" is a person's name that must be redacted, and when "hunter" refers to a tick-borne virus, preserving the medical data while protecting the patient.

Algorithmic Bias in Clinical NER

BERT is incredibly smart, but it learned everything it knows by reading historical human text. This means it also learned our biases.

The Representation Gap Clinical datasets are historically skewed. If BioBERT is trained primarily on medical notes from specific demographics, it becomes incredibly good at extracting symptoms expressed in standard clinical English, but it might fail to extract symptoms expressed in different cultural dialects or vernaculars.

The Danger in Mental Health If an NER model is used to flag patients at risk of severe depression, but it fails to recognize the implicit vocabulary used by a specific minority group, that group gets systematically under-diagnosed by the algorithm.

- A high F1-Score doesn't matter if the model only works for half the population!

The Post-BERT Era (Modern Day)

BERT dominated from 2018 to about 2022, but with LLM NER task shifted.

The End of BIO Tags? For decades, NER was a "Sequence Labeling" task (guessing B-PER, I-PER for every single word). Modern LLMs are text generators. They don't use BIO tags.

Zero-Shot and Few-Shot Prompting Instead of fine-tuning a BERT model on 10,000 medical records, we can now just write a prompt to an LLM:

- *"Read this doctor's note and output a JSON list of all the medications and dosages."* * The model just *generates* the list. It can extract entities it was never explicitly trained to look for (Zero-Shot NER).

The Trade-off Generative models are incredibly flexible and require no training data. However, for strict, highly secure, offline hospital environments, a small, fine-tuned ClinicalBERT is still often cheaper, faster, and more predictable than a massive generative LLM.

Summary: *From Memorizing Words to Understanding Meaning*

Era 1: The Dictionary Phase (Regex & Rules)

- *The Approach:* "Memorize every medical term and city name."
- *The Flaw:* Language is too ambiguous. A dictionary doesn't know the difference between "Apple" the fruit and "Apple" the company.

Era 2: The Sequential Phase (HMMs, CRFs, & BiLSTMs)

- *The Approach:* "Read the sentence left-to-right to find clues, and use math to enforce grammar rules."
- *The Flaw:* It struggled with long paragraphs, implicit meanings, and complex logical pronoun shifts.

Summary: *From Memorizing Words to Understanding Meaning*

Era 3: The Transformer Phase (Self-Attention & BERT)

- *The Approach:* "Look at every word simultaneously. Dynamically change the math of a word based on the neighbors surrounding it."
- *The Victory:* Deep context. The AI finally gained implicit "world knowledge," allowing it to read messy doctor's notes and nuanced mental health forums with human-level accuracy.