What is an Agent? A brief history of language modeling

Shreya Havaldar

Introductions

Shreya

Davis (TA)

Vitoria (TA)

A little about me!

- From LA + Orange County
- Graduated from USC!
- Started my PhD at Penn in 2021
- I research sociotechnical alignment of LLMs (how personal + cultural context influence what a “correct generation” should be

My mom + my dog!

Welcome!

A few things to note

1. This is the first time this class is being offered
2. This is also my first time teaching
3. There will (inevitably) be some bumps in the road! We will work them out!!

This course (like all mini-courses) **is meant to be fun!** It will be equal parts computer science and design.

Introductions

1. Name
2. Year + major
3. Highlight of your winter break
4. Favorite way to use AI agents

Welcome!

What you'll get from this course

1. A basic understanding of how LLMs work
2. Overview of agentic design principles
3. Insights into how agents are being used in academia + industry
4. Hands-on project where you build an agent!

Grading

Grades will be 35% homeworks, 45% final project, and 20% participation.

- HWs are individual
- Final project is in groups of 2
- Participation is based on attendance + contribution during class activities
    - You are allowed to miss class 2 times without penalty (otherwise you need to email me)

Policies

- AI tools are allowed for coding tasks, provided **you fully understand and can explain any code you submit.**

- AI delegation is NOT allowed for writing components of assignments.
  - Short-answer questions in the HWs
  - Design document for final project
  - Note: collaboration != delegation

- You have 3 late days you can use as needed

Feedback

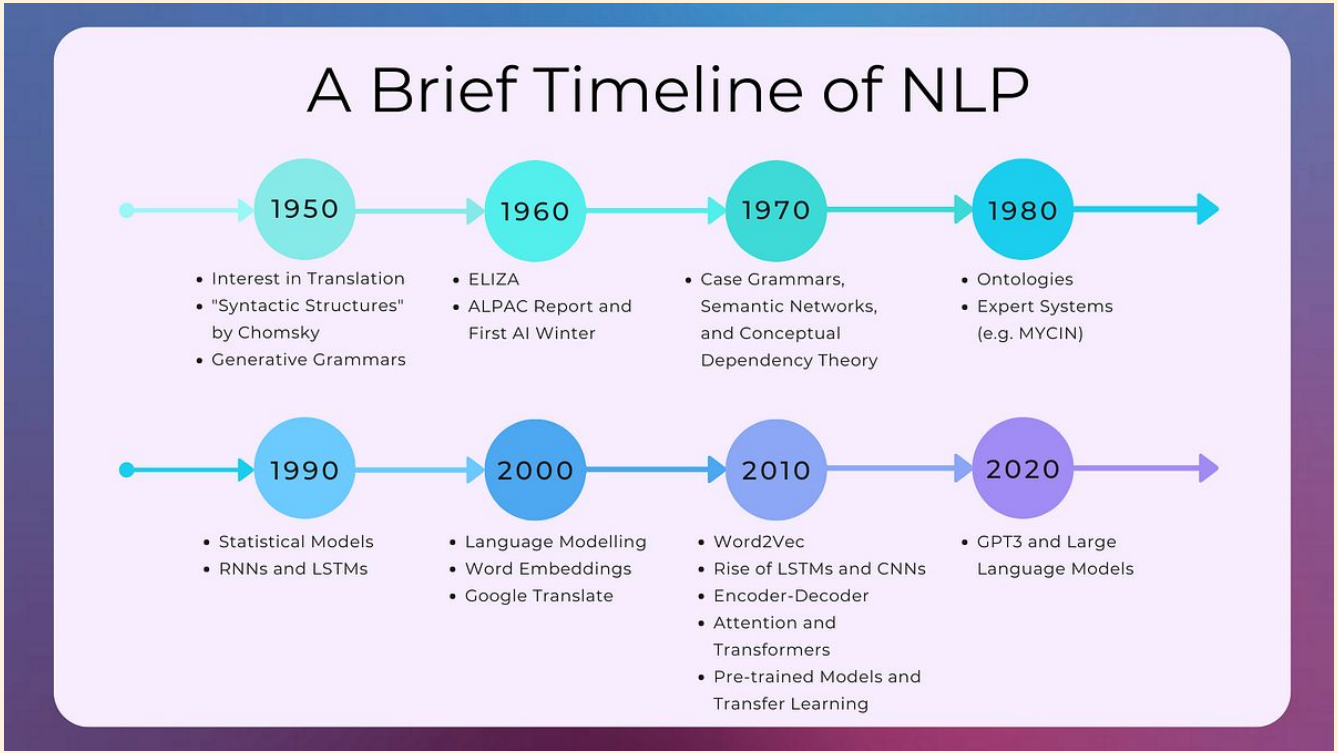
This is a pilot course! We would like to offer it again, so feedback is welcome and encouraged

- Myself or the TAs
- Mini-course facilitators (Swapneel Sheth, Harry Smith, or Kyrie Dowling)

Today’s Lecture

1. **History of language modeling**
2. Defining the AI Agent

The Evolution of language models

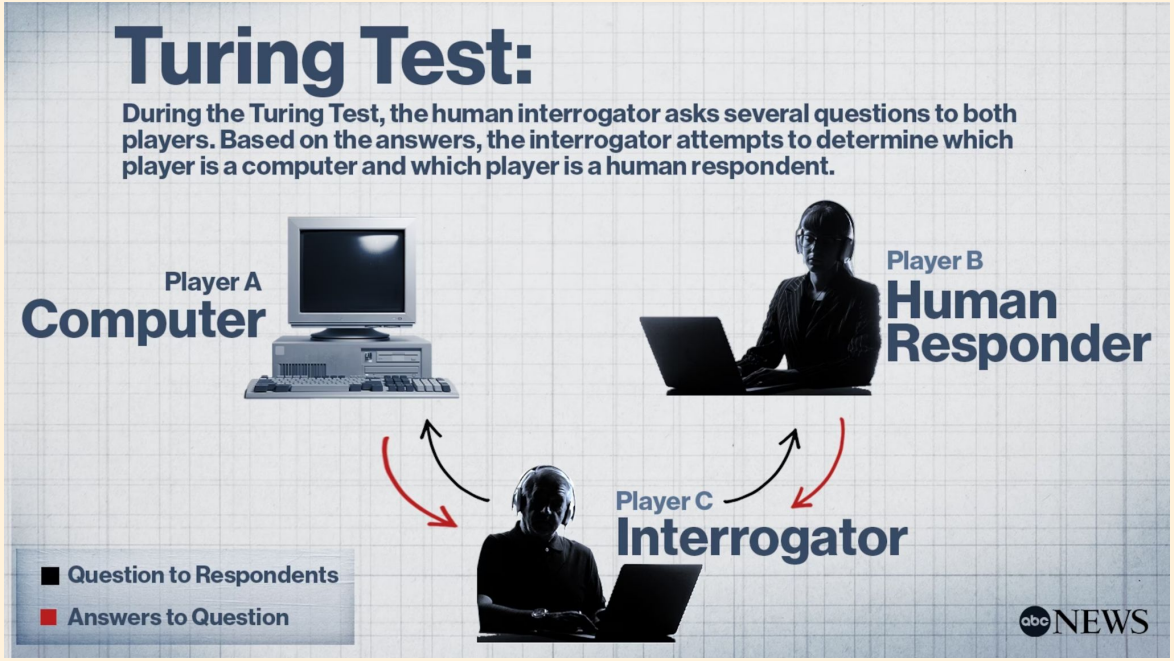


The Rule-Based Era (1950s – 1990s)

Early researchers viewed language as a set of logical puzzles. If you could map every grammatical rule, you could "solve" language.

The Turing Test (1950)

Alan Turing proposed that if a machine could mimic human conversation perfectly, it could be considered "thinking."



Chomskyan Linguistics

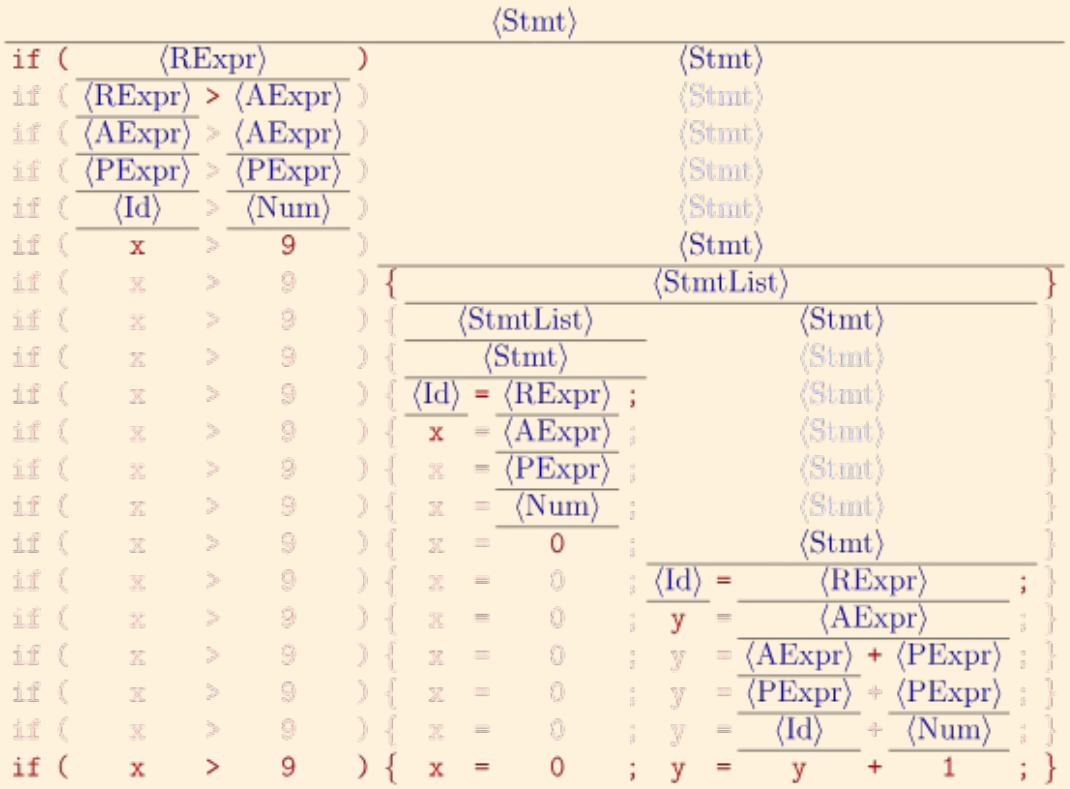
Noam Chomsky argued for "Universal Grammar"—the idea that humans have an innate, rule-based biological structure for language.

Computers tried to mimic this with **Context-Free Grammars (CFGs)**.

CFGs

A formal system in computer science that uses recursive rules to describe the structure of languages.

⟨Stmt⟩ → ⟨Id⟩ = ⟨RExpr⟩ ;
⟨Stmt⟩ → { ⟨StmtList⟩ }
⟨Stmt⟩ → if (⟨RExpr⟩) ⟨Stmt⟩
⟨StmtList⟩ → ⟨Stmt⟩
⟨StmtList⟩ → ⟨StmtList⟩ ⟨Stmt⟩
⟨RExpr⟩ → ⟨RExpr⟩ > ⟨AExpr⟩
⟨RExpr⟩ → ⟨RExpr⟩ < ⟨AExpr⟩
⟨RExpr⟩ → ⟨RExpr⟩ >= ⟨AExpr⟩
⟨RExpr⟩ → ⟨RExpr⟩ <= ⟨AExpr⟩
⟨RExpr⟩ → ⟨AExpr⟩
⟨AExpr⟩ → ⟨AExpr⟩ + ⟨PExpr⟩
⟨AExpr⟩ → ⟨AExpr⟩ - ⟨PExpr⟩
⟨AExpr⟩ → ⟨PExpr⟩
⟨PExpr⟩ → ⟨Id⟩
⟨PExpr⟩ → ⟨Num⟩
⟨Id⟩ → x
⟨Id⟩ → y
⟨Num⟩ → 0
⟨Num⟩ → 1
⟨Num⟩ → 9



Activity: Rules for WSD

WSD = **Word Sense Disambiguation**

For example, does “date” mean

(1) a dried fruit

(2) a romantic meeting

(3) a calendar day

Get into groups of 2!

Activity: Rules for WSD

Phase 1: Build Your “Date” Rules (15 Minutes)

Write strict rules **that ONLY using logic**. Examples:

IF "love" is in the sentence, THEN it is a romantic meeting.

IF "january" is in the sentence, THEN it is a calendar day.

```
def disambiguate_date(input_sentence):
    …
    return {“dried_fruit”, “romantic date”, “calendar_day”}
```

Activity: Rules for WSD

Phase 2: Break someone else's "Date" Rules (5 + 5 Minutes)

IF "love" is in the sentence, THEN it is a romantic meeting.
IF "january" is in the sentence, THEN it is a calendar day.

→ *I have a Hinge **date** planned in **January***
→ *I **love** to put **dates** in my smoothie*

Feel free to use AI tools!

The Statistical Shift & Word Embeddings (1990s – 2013)

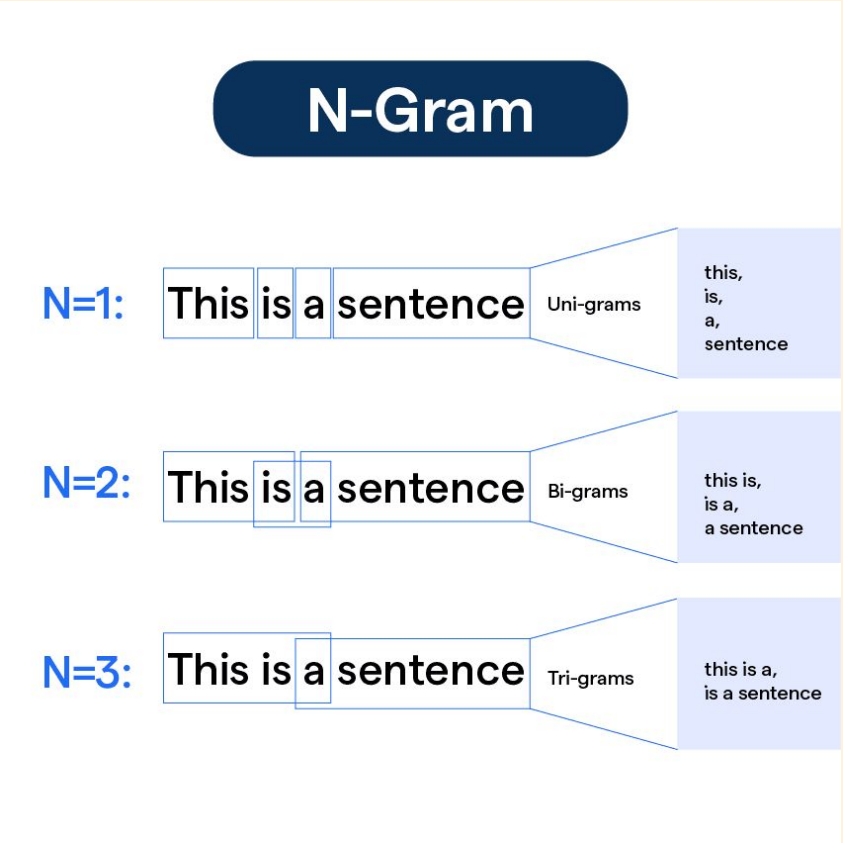
The world is too messy for rules!

Instead of rules, researchers began using large corpora (like the Wall Street Journal, Wikipedia, etc.) to calculate the probability of word sequences.

N-Grams

Predicting the n^{th} word based on the previous ***n-1*** words.

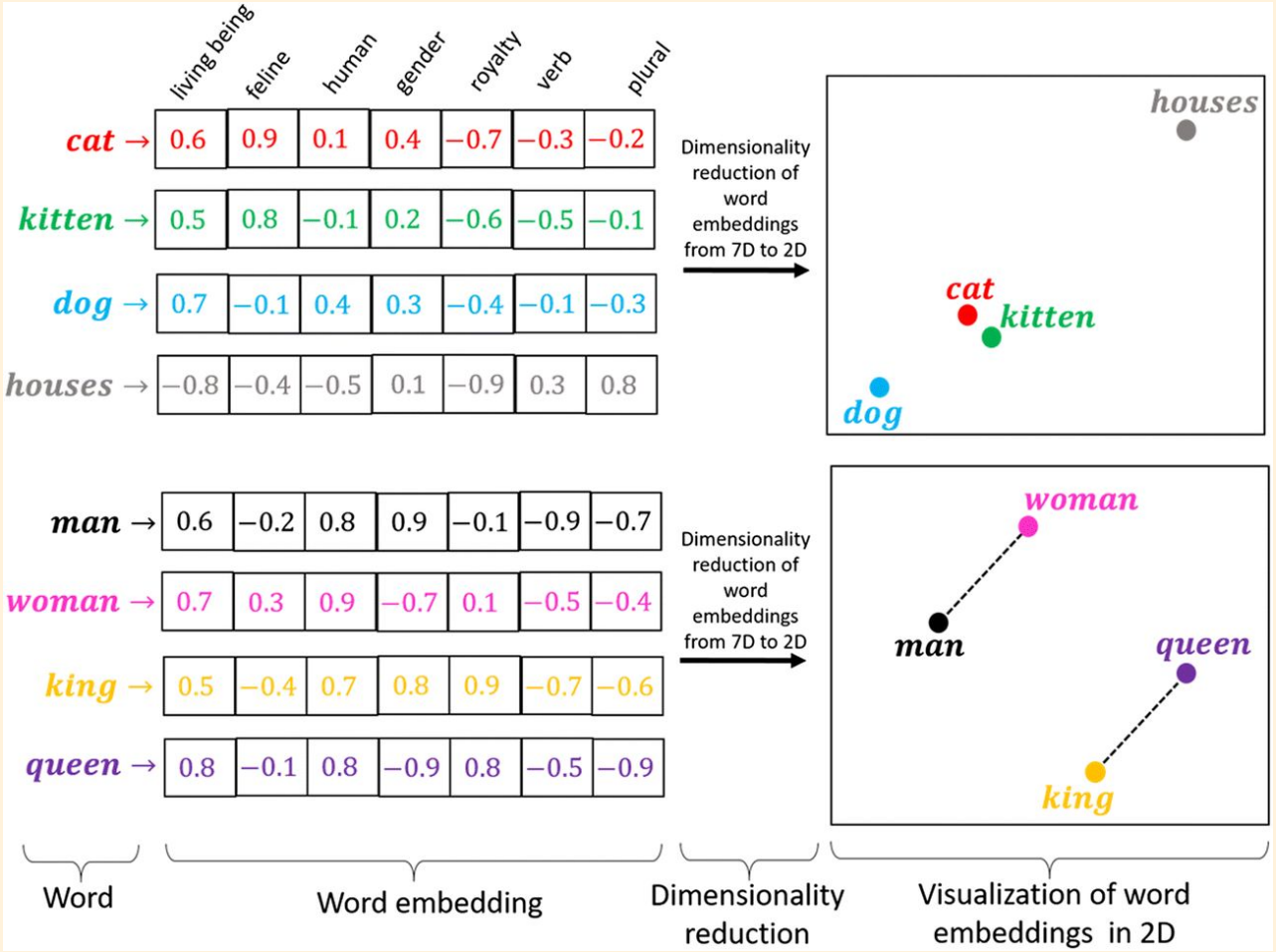
This is fundamentally a **Markov Chain** — the future state depends only on the current state, not the entire past history



The Vector Revolution (2013)

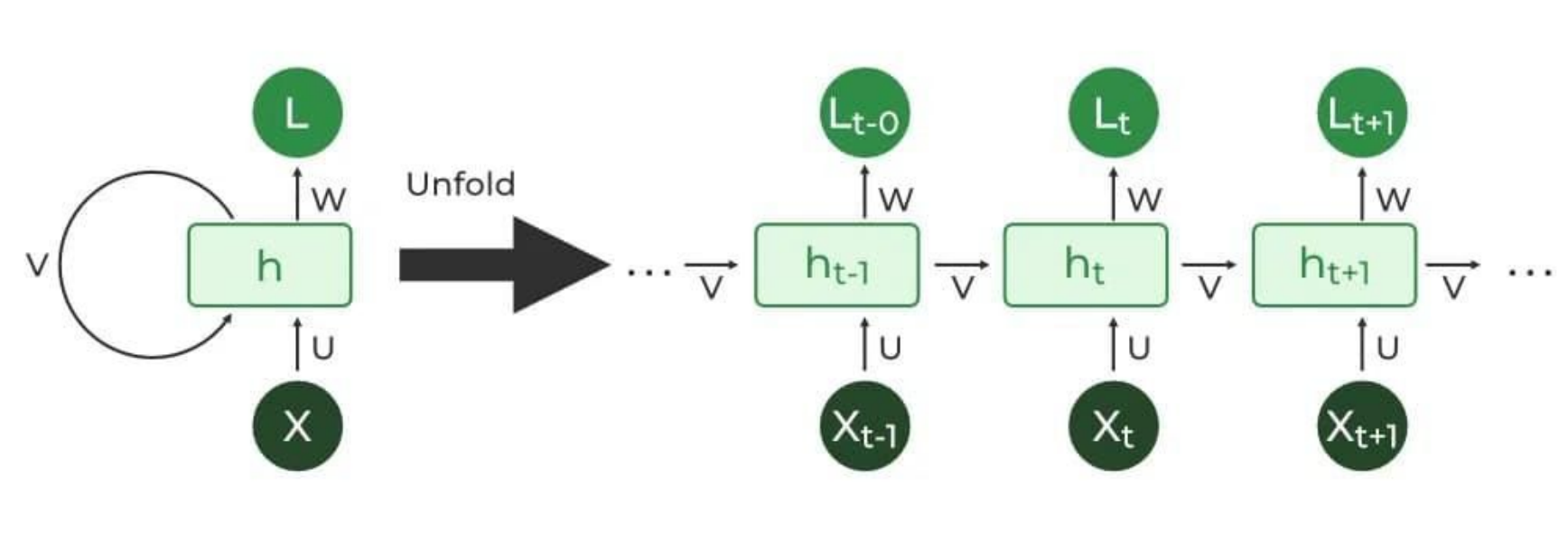
The introduction of **Word2Vec** and **GloVe**. Instead of representing the word "Apple" as a unique ID, we represented it as a point in a 300-dimensional space.

Key Insight: *"You shall know a word by the company it keeps."*
→ Words with similar meanings (e.g., "King" and "Queen") were mathematically closer together.


living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

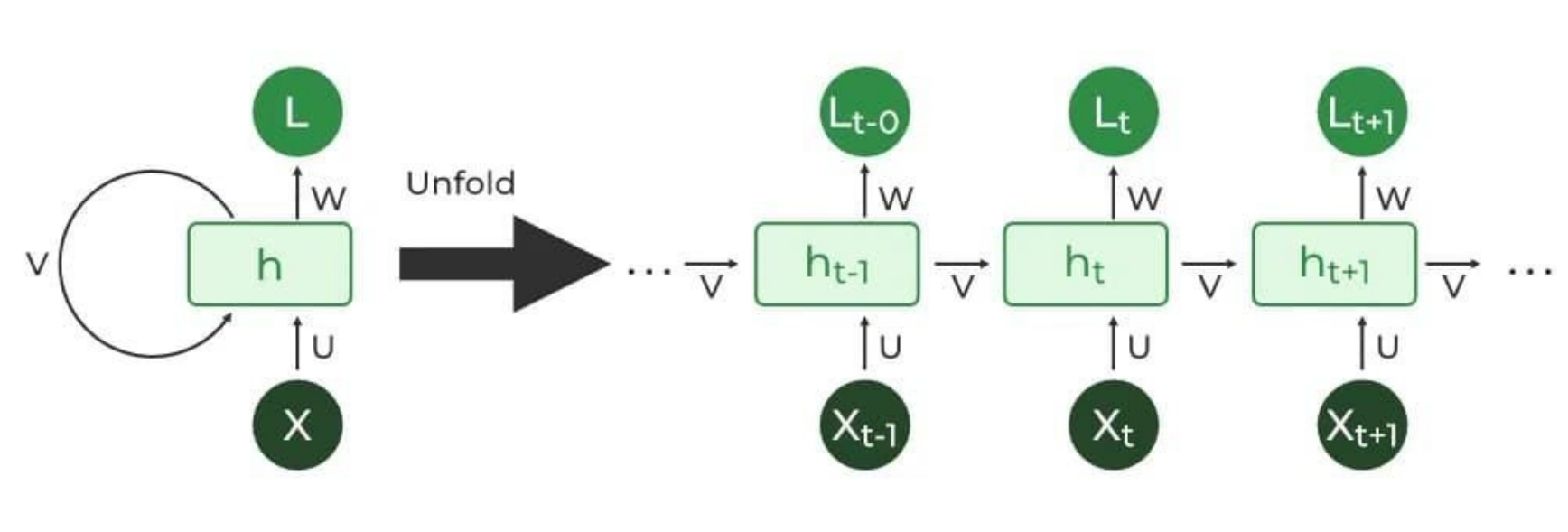
Recurrent Neural Networks (RNNs)

RNNs process tokens one by one, maintaining a "hidden state."



Recurrent Neural Networks (RNNs)

The Problem: **The "Vanishing Gradient."** As the input sentence grows, the influence of the first word on the last word decays to zero.

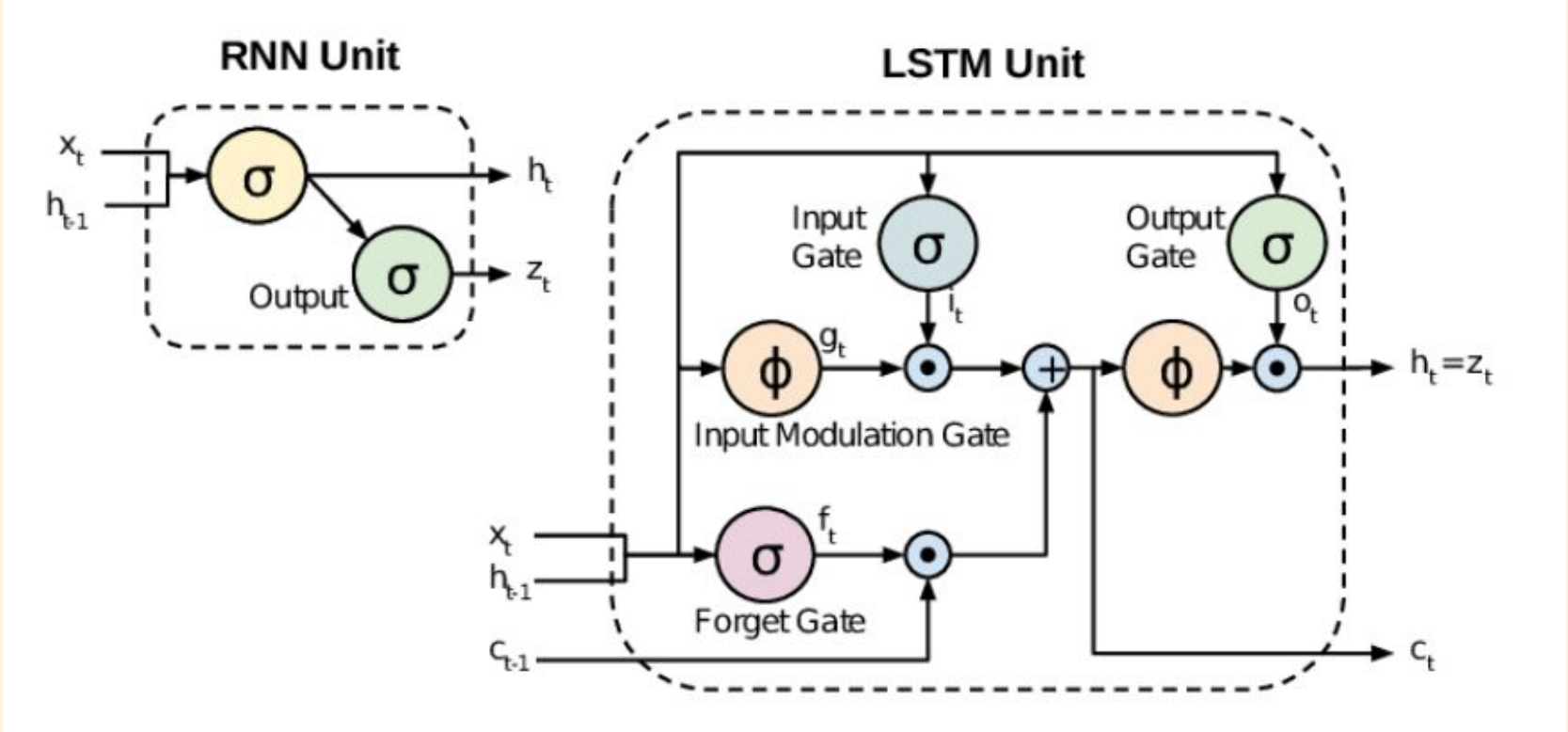


LSTMs (Long Short-Term Memory)

Introduced **gates** and and a **cell state**

- Gates control the flow of information into and out of the cell state
- Input, Forget, and Output gates
- Cell state acts as a persistent memory conveyor belt.

Solved the vanishing gradient problem!


RNN Unit
x_t
h_{t-1}
σ
h_t
Output
σ
z_t
LSTM Unit
Input Gate
σ
i_t
Output Gate
σ
o_t
g_t
ϕ
Input Modulation Gate
ϕ
$h_t = z_t$
x_t
h_{t-1}
σ
f_t
Forget Gate
c_{t-1}
c_t

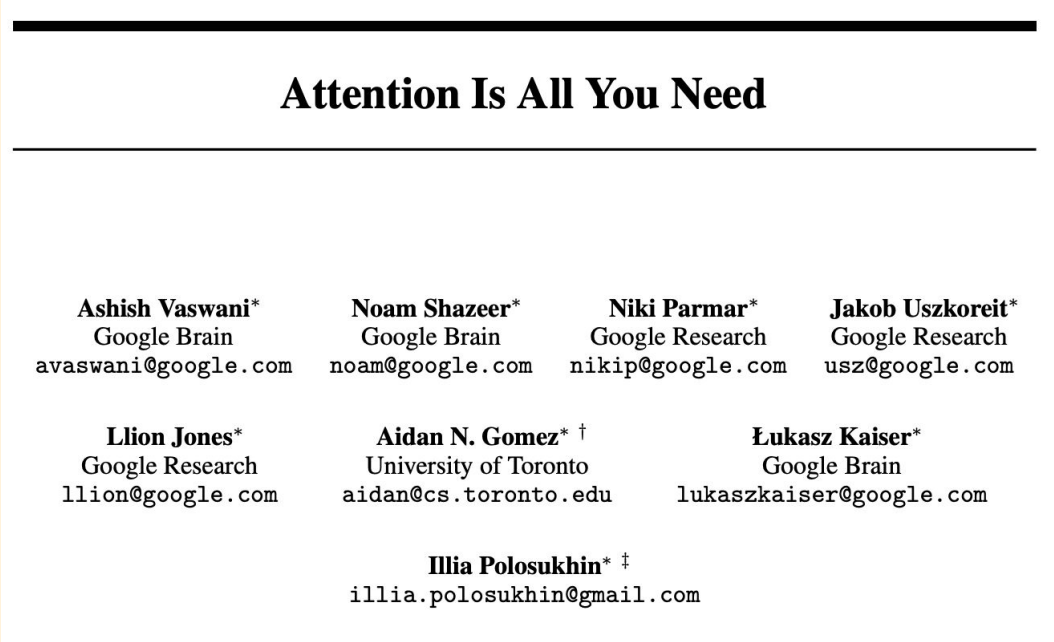
LSTMs (Long Short-Term Memory)

New problem: Still **sequential**.

- You cannot parallelize the training.
- You must wait for word #1 to finish before starting word #2.

The Transformer & Scaling Laws (2017 – Present)

The paper **"Attention is All You Need"** removed the sequence requirement entirely.



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

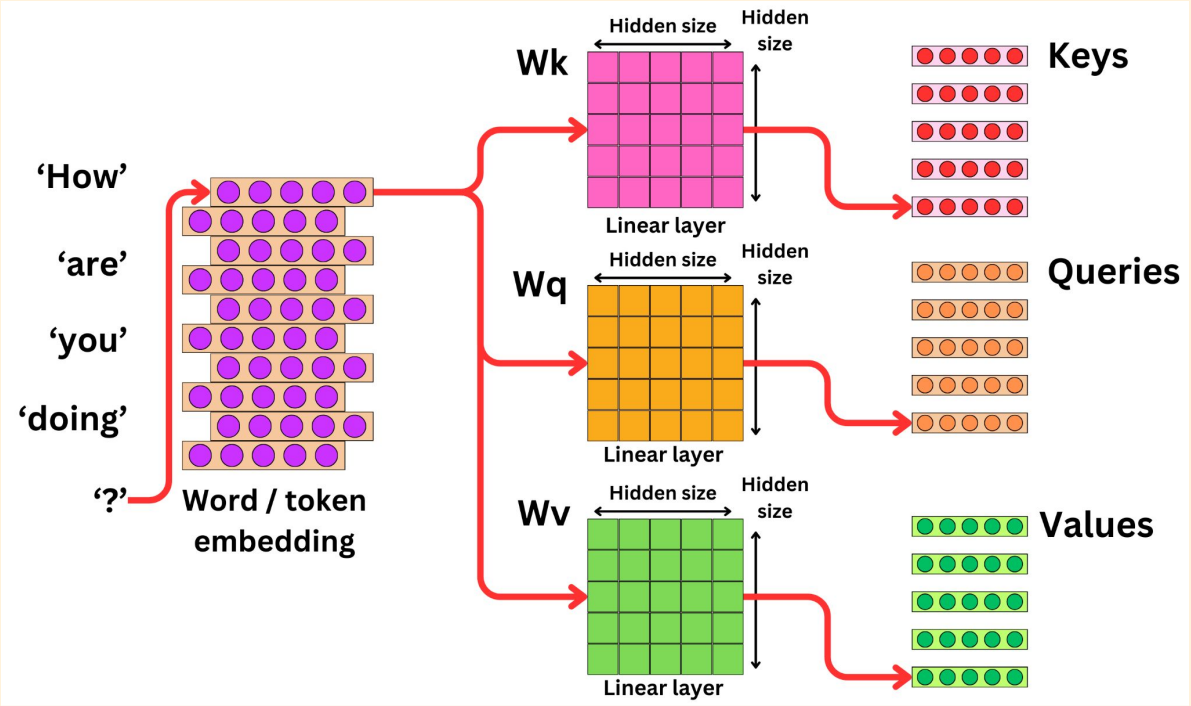
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

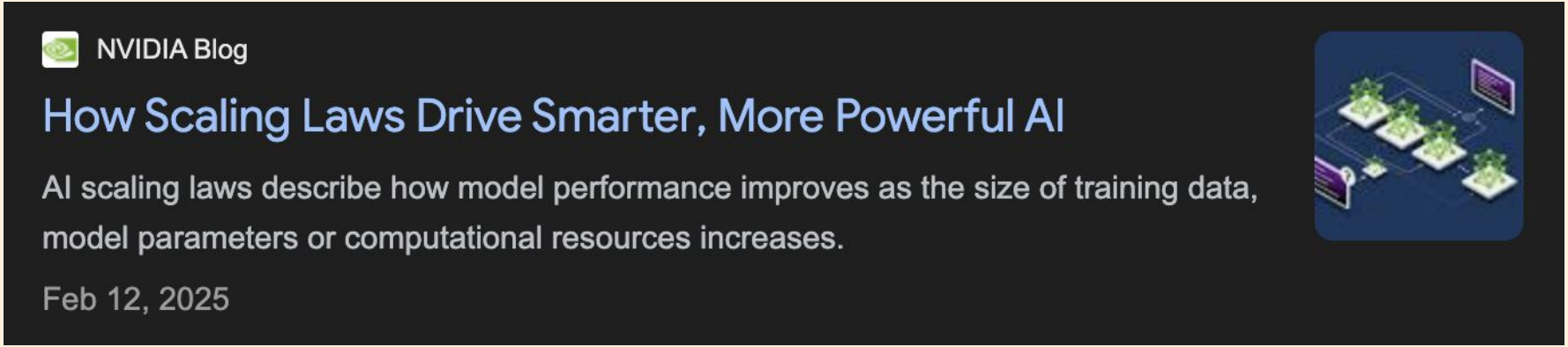
The Transformer

Self-Attention: Every word in a sentence looks at every other word at the same time. This allows the model to understand global context instantly.



Scaling Laws

Researchers discovered that as you increase the number of parameters and the amount of data, the "Loss" (error) decreases in a predictable power law. This led to the race for trillion-parameter models.



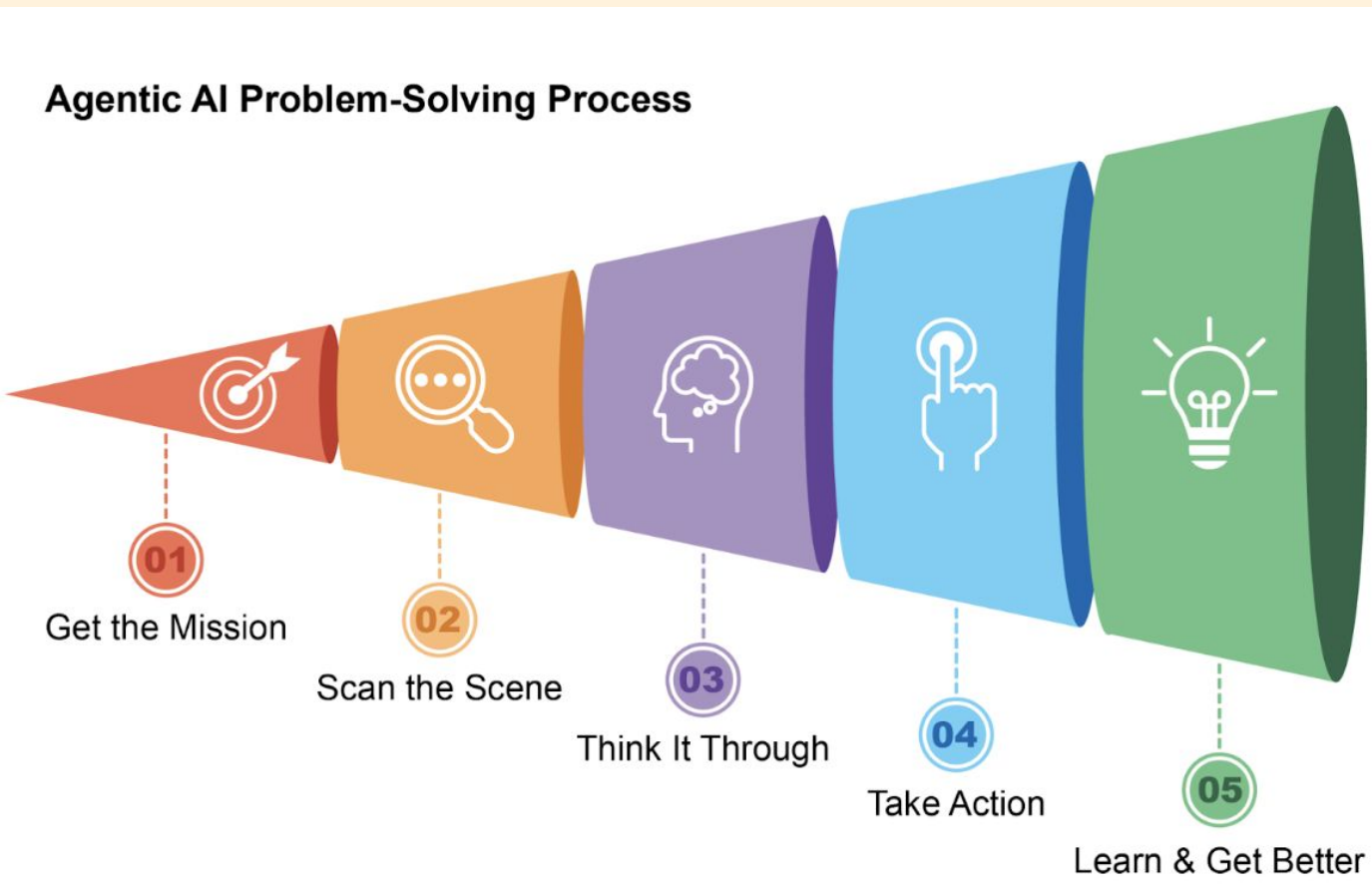
Today’s Lecture

1. History of language modeling
2. **Defining the AI Agent**

From Completion to Agency

Passive LLM: You give it a prompt, and it gives you a completion. It has no memory of the world and no way to change it.

Active Agent: An agent is Autonomous, Reactive, and Proactive. It can observe a result, realize it made a mistake, and try a different approach.


Agentic AI Problem-Solving Process
01
Get the Mission
02
Scan the Scene
03
Think It Through
04
Take Action
05
Learn & Get Better

The Cognitive Architecture of an Agent

1. **Planning**
2. **Memory**
3. **Tool Use / Action**
4. **Perception**

The Cognitive Architecture of an Agent

1. **Planning**
    a. Task Decomposition: The agent breaks a "Goal" into "Steps."
    b. Self-Reflection: The agent looks at its own output and asks, "Does this make sense?" (e.g., the Self-Refine or Reflexion frameworks).
    c. Chain of Thought (CoT): Forcing the model to "show its work" step-by-step, which significantly reduces logic errors.

The Cognitive Architecture of an Agent

1. Planning

2. **Memory**
    a. Short-term: The conversation history stored in the context window.
    b. Long-term: Vector Databases (using RAG or other retrieval mechanisms). The agent can "search" its past experiences or external documents to inform its current action.

The Cognitive Architecture of an Agent

1. Planning
2. Memory

3. **Tool Use / Action**
    a. Function Calling: The model doesn't just write text; it outputs a specific piece of code that a computer can execute.
    b. The MRKL (Miracle) Architecture: *Modular reasoning, knowledge, and language.*
        i. A system that combines an LLM with "expert" tools (like a calculator or a weather API).

The Cognitive Architecture of an Agent

1. Planning
2. Memory
3. Tool Use / Action

4. **Perception**
    a. Modern agents are often Multimodal. They don't just "read" the internet; they can "see" a screenshot of a website or "hear" a user's voice command.
    b. Inputs from different modalities can be processed together to give the agent a better understanding of its task

The ReAct Pattern (Reasoning + Acting)

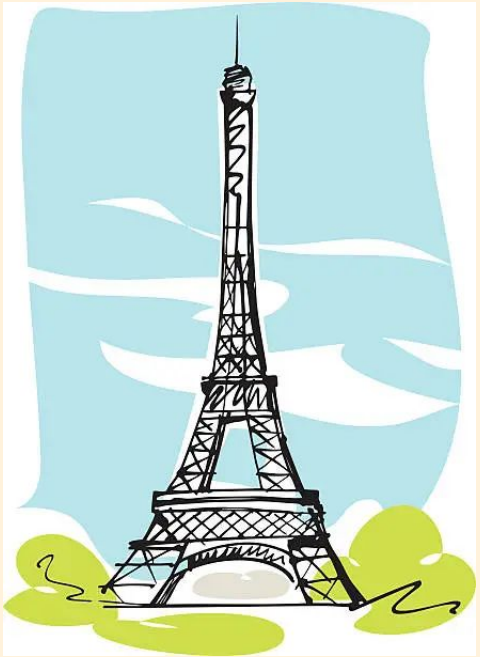
The most common agentic loop is ReAct:

Example: “How many women are there in Paris?”

Step 1 (Thought): "I need to find the population of Paris and divide it by 2."
Step 2 (Action): Search("Population of Paris")
Step 3 (Observation): "The search result says 2.1 million."
Step 4 (Thought): "Now I need to use the calculator tool."
Step 5 (Action): Calculate(2100000 / 2)

Activity: Applying ReAct (15 min)

Get into groups of 2. Pick a complex, open-ended problem (e.g. plan me a 1-week trip to Paris on a 500 euro budget)



Sketch out how an agent would approach this problem

1. **Planning:** How does the agent break this down?
2. **Tool Use:** Which APIs/external tools are needed?
3. **Memory:** What does it need to "retrieve" from the user's past data to succeed?
4. **Perception:** What real-time data must it "observe" to make a decision?

Updates

- There are 100+ people on the waitlist! If you plan to drop, please lmk sooner rather than later

- Reminder to sign up for EdStem + Gradescope

- HW 1 is on website and is due in 2 weeks!

- Office hours will be in AGH (times to be posted on website this week). **Email me if you don’t have access to AGH. All SEAS students should have access.**