Language Model Architecture

Yu Meng
University of Virginia
yumeng5@virginia.edu

Jan 14, 2025

(Recap) Course Information & Logistics

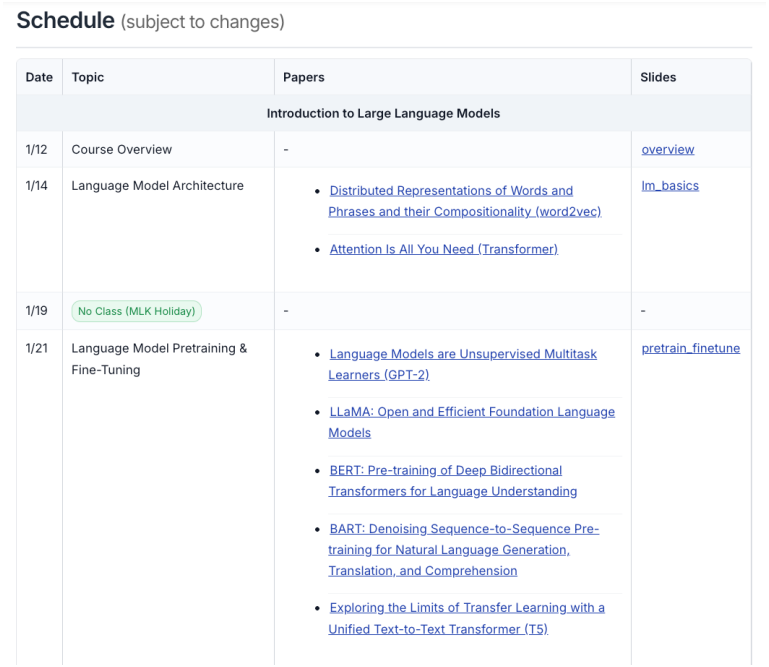
- Instructor: **Yu Meng** (yumeng5@virginia.edu)
- TAs:
  - **Wei-Lin Chen** (wlchen@virginia.edu)
  - **Zhepei Wei** (zhepei.wei@virginia.edu)
  - **Xinyu Zhu** (xinyuzhu@virginia.edu)
- Time: Mondays & Wednesdays 2:00pm - 3:15pm
- Location: Olsson Hall 009
- Office Hour: Mondays & Wednesdays after class
- We'll use Piazza (accessible via Canvas) to answer logistics/technical questions

(Recap) Course Information & Logistics

- This course is designed to be a **research-oriented graduate-level** course
- Seminar-style: a substantial focus on reading, presenting and discussing important papers and conducting research projects
- A comprehensive overview of cutting-edge developments in NLP
- Prerequisites: CS 4770 or CS 4774 (having deep learning background is important!)
- This course may benefit you if
  - You are working on NLP research (PhD/MS research students)
  - Your research uses NLP models/tools
  - You aim for a job that involves using NLP models/tools
  - You are very interested in the cutting-edge topics of NLP and willing to spend time to learn

(Recap) Course Format & Grading

- Course Website: https://yumeng5.github.io/teaching/2026-spring-cs6770



Schedule (subject to changes)

Date	Topic	Papers	Slides
		Introduction to Large Language Models	
1/12	Course Overview	-	overview
1/14	Language Model Architecture	• Distributed Representations of Words and Phrases and their Compositionality (word2vec) • Attention Is All You Need (Transformer)	lm_basics
1/19	No Class (MLK Holiday)	-	-
1/21	Language Model Pretraining & Fine-Tuning	• Language Models are Unsupervised Multitask Learners (GPT-2) • LLaMA: Open and Efficient Foundation Language Models • BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding • BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension • Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)	pretrain_finetune

(Recap) Course Format & Grading: Paper Presentation (30%)

- Starting from the 4th lecture **(1/26)**, each lecture will be presented by a group of 2 or 3 students
- Every group presents one lecture (3 papers)
- Signup sheet: https://docs.google.com/spreadsheets/d/1MnapwDyIdG9Z5LyQ0QwojV85L4lYirVd_Y6CPmFFjmU/edit?usp=sharing
- You can sign up for the topic you are interested in – slots are first come, first served!
- The dates listed on the course website are subject to change – please sign up based on the topic rather than the date

(Recap) Course Format & Grading: Paper Presentation (30%)

- **Presentation duration**: strictly limited to 60 minutes, followed by a 10-minute question-and-answer session with the audience & instructor
- **Deadline**: Email your slides to the instructor and TAs at least 48 hours before your presentation (e.g., if presenting on Monday, slides should be emailed by Saturday 2pm)
- You will receive feedback from the instructor to improve your slides (if necessary, the instructor may schedule a meeting with your team to go over the slides)
- Late submissions result in a 50% presentation grade deduction
- Detailed grading rubrics and tips can be found on the course website
- First three student lectures automatically receive **5%, 3%, 1% extra credit of final grade**

(Recap) Course Format & Grading: Participation (20%)

- Starting from the 4th lecture **(1/26)**, everyone is required to complete two mini-assignments
- **Pre-lecture question**: read the 3 papers to be introduced in the lecture, and submit a question you have when you read them
- **Post-lecture feedback**: provide feedback to the presenters after the lecture
- We'll use Google Forms to collect pre-lecture questions and post-lecture feedback and share them with the presenters
- **Deadlines**: pre-lecture questions are due one day before the lecture (e.g., For Monday lectures, you need to submit the question by Sunday 11:59 pm); post-lecture feedback is due each Friday (both Monday & Wednesday feedback is due Friday 11:59 pm)
- Lectures are not recorded, but slides will be posted on the course website

(Recap) Course Format & Grading: Project (50%)

- Complete a research project, present your results, and submit a project report
- Work in a team of 2 or 3 (any other team size requires prior approval from the instructor) – may or may not be the same team as your presentation group
- (Type 1) A comprehensive survey report: carefully examine and summarize existing literature on a topic covered in this course; provide detailed and insightful discussions on the unresolved issues, challenges, and potential future opportunities within the chosen topic
- (Type 2) A hands-on project: not constrained to the course topics but must be centered around NLP; doesn't have to involve large language models (e.g., train or analyze smaller-scale language models for specific tasks); eligible for extra credits if publishable
- **Project proposal**: 5% (ddl: 2/6); **Mid-term report**: 10% (ddl: 3/13); **Final presentation** (ddl: 4/20) **and final report**: 35% (ddl: 5/5)

Overview of Course Contents

- Introduction to Large Language Models
- Reasoning with Language Models
- Knowledge & Factuality
- Efficiency
- Language Model Alignment
- Language Agents
- Evaluation and Ethical Considerations of Language Models
- Looking Forward

Parametric Knowledge

Language models can be prompted for factual question answering

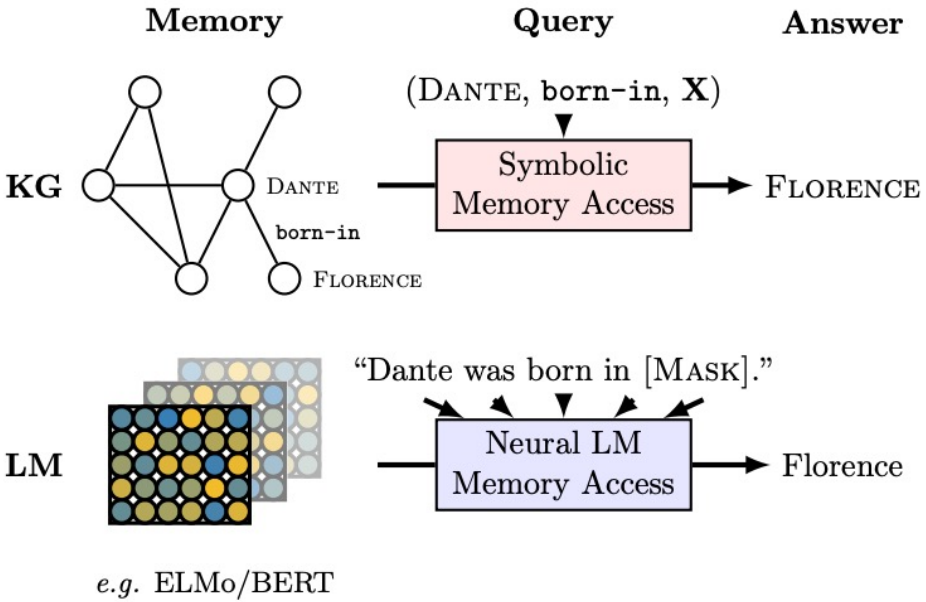


Figure source: https://arxiv.org/pdf/1909.01066.pdf

Retrieval-Augmented Generation (RAG)

Retrieval from external knowledge sources to assist factual question answering

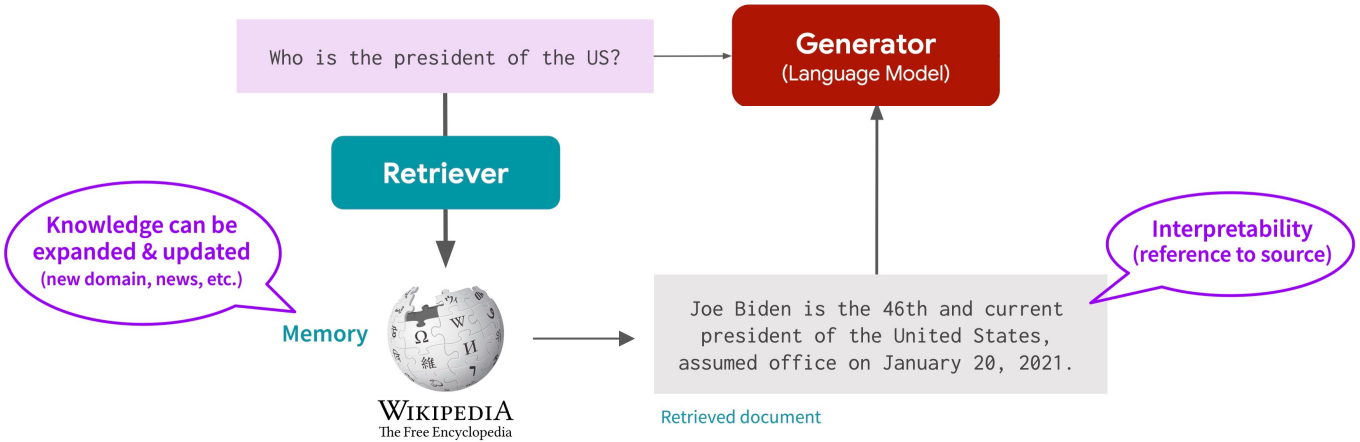


Figure source: https://cs.stanford.edu/~myasu/blog/racm3/

Long-Context Issues

U-shaped performance curve under long context: LLMs are better at using relevant information that occurs at the very beginning (**primacy bias**) or end of its input context (**recency bias**)

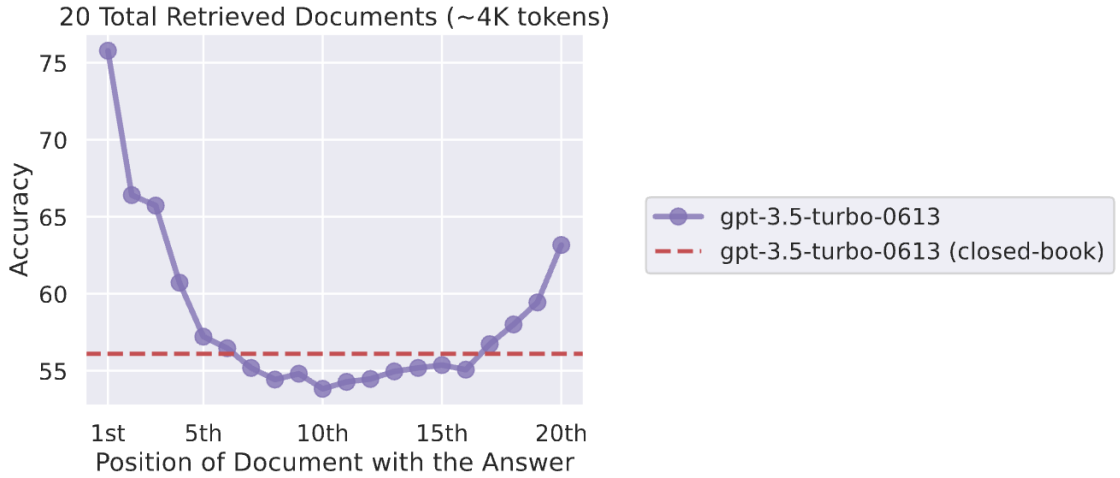


Figure source: https://arxiv.org/pdf/2307.03172

Overview of Course Contents

- Introduction to Large Language Models
- Reasoning with Language Models
- Knowledge & Factuality
- Efficiency
- Language Model Alignment
- Language Agents
- Evaluation and Ethical Considerations of Language Models
- Looking Forward

Efficient Architectures

State space models (e.g., Mamba) achieves linear-time complexity with Transformer-level quality for sequence modeling

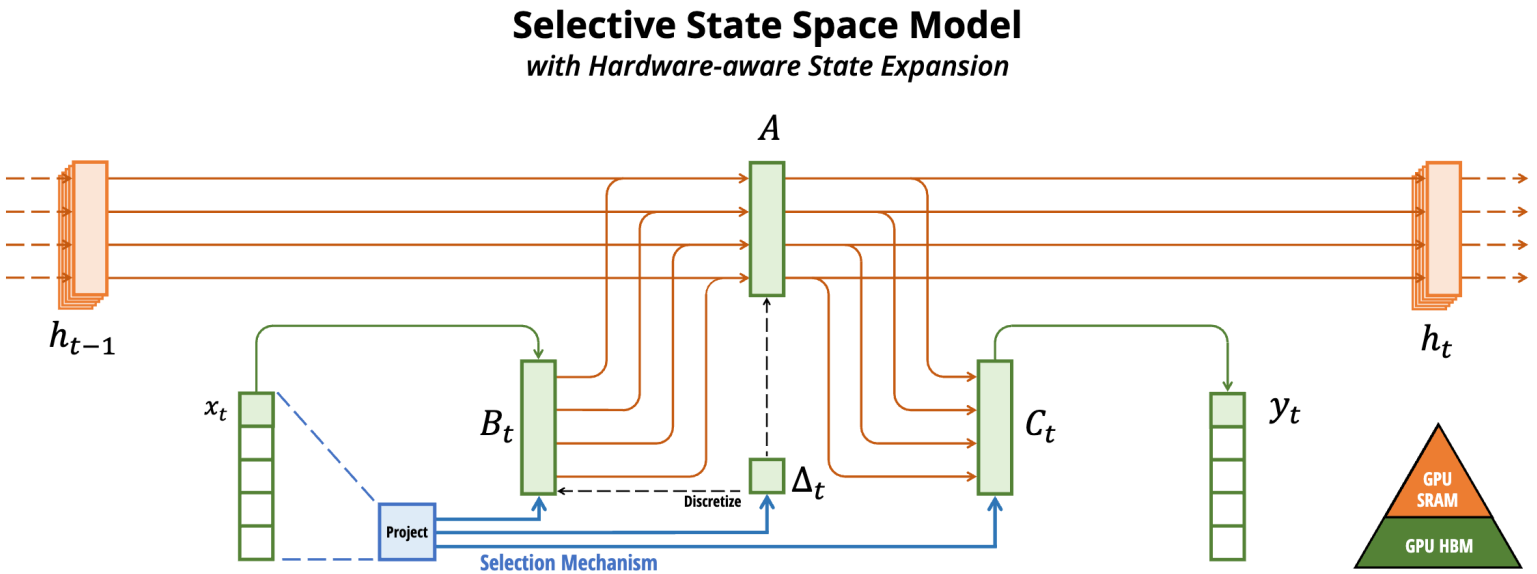


Figure source: https://arxiv.org/pdf/2312.00752

Sparse Models

Only one expert is activated for each token

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as
- (tokens_per_batch / num_experts) * capacity_factor
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.

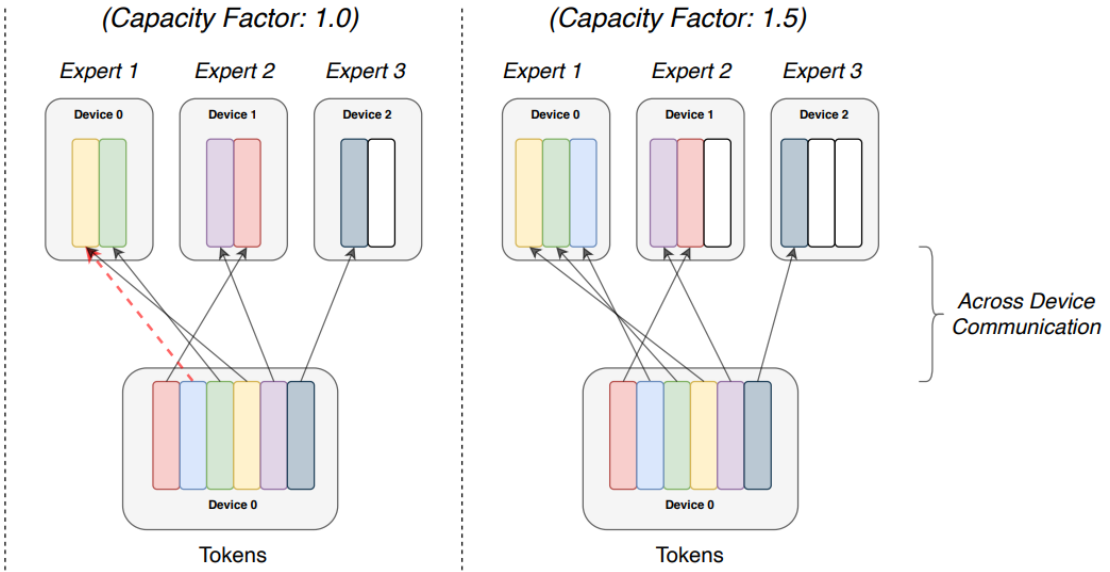


Figure source: https://arxiv.org/pdf/2101.03961.pdf

Overview of Course Contents

- Introduction to Large Language Models
- Reasoning with Language Models
- Knowledge & Factuality
- Efficiency
- Language Model Alignment
- Language Agents
- Evaluation and Ethical Considerations of Language Models
- Looking Forward

Aligning Language Models for Instruction Following

Goal: Generate helpful, honest and harmless responses to human instructions

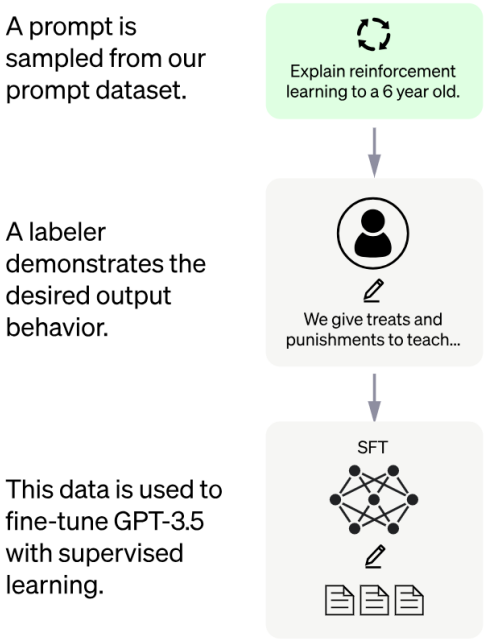


Figure source: https://openai.com/blog/chatgpt

Reinforcement Learning from Human Feedback (RLHF)

Further learning from pairwise data annotated by humans

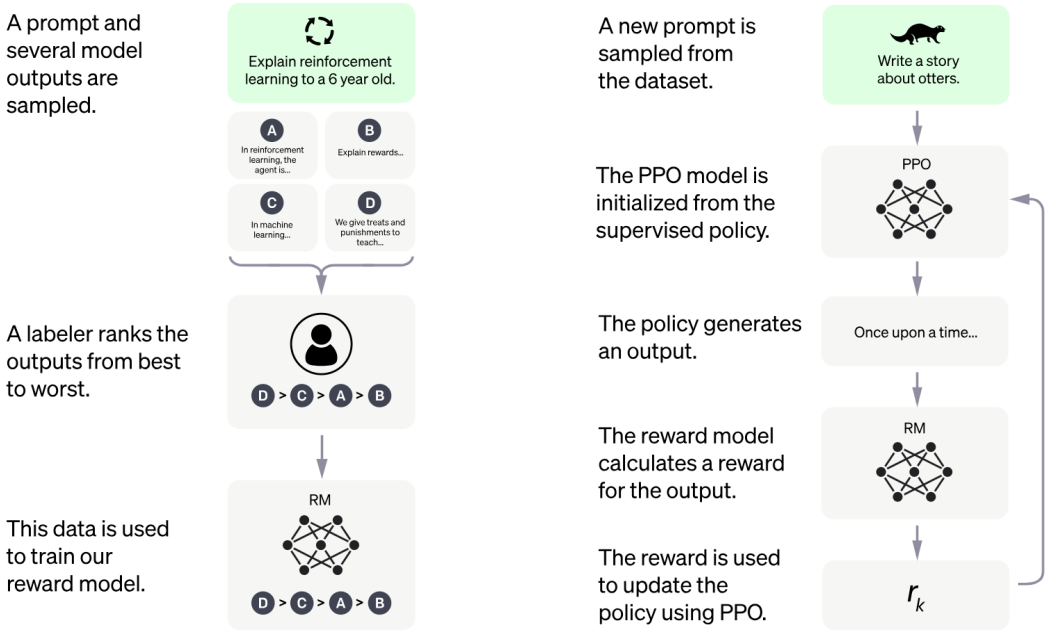


Figure source: https://openai.com/blog/chatgpt

Overview of Course Contents

- Introduction to Large Language Models
- Reasoning with Language Models
- Knowledge & Factuality
- Efficiency
- Language Model Alignment
- Language Agents
- Evaluation and Ethical Considerations of Language Models
- Looking Forward

Claude 3.5: Computer Use

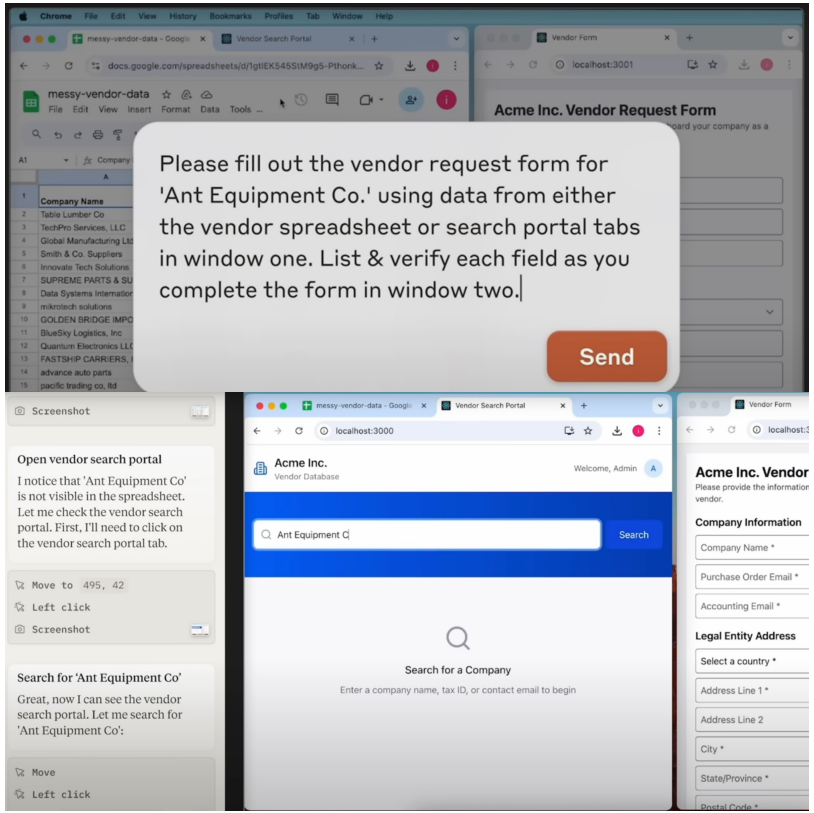
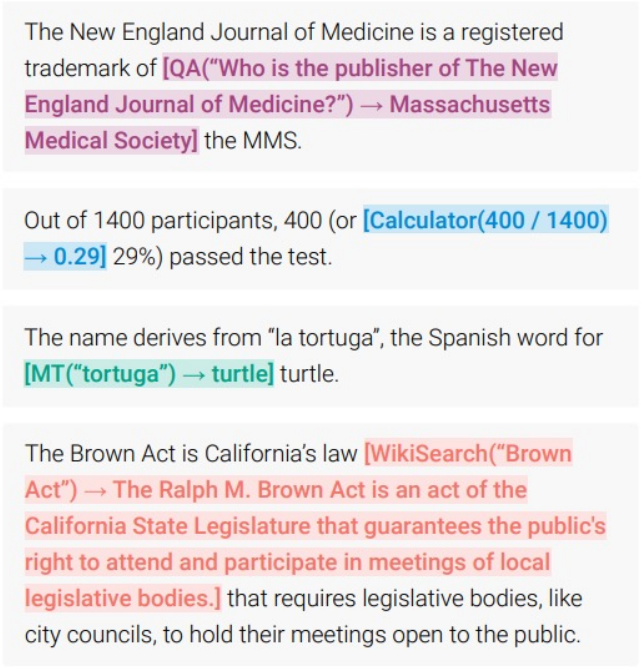


Figure source: https://www.anthropic.com/news/3-5-models-and-computer-use

Language Model Agents: Tool Usage

Task execution assisted with external tools



Question answering system

Calculator

Machine translation system

Wikipedia search

Figure source: https://arxiv.org/pdf/2302.04761.pdf

Language Model Agents: Coding

LLMs can be trained for code writing

Docstring Generation

```
def count_words(filename: str) -> Dict[str, int]:
    """
    Counts the number of occurrences of each word in the given file.

    :param filename: The name of the file to count.
    :return: A dictionary mapping words to the number of occurrences.
    """
    with open(filename, 'r') as f:
            word_counts = {}
            for line in f:
                for word in line.split():
                    if word in word_counts:
                        word_counts[word] += 1
                    else:
                        word_counts[word] = 1
        return word_counts
```

Multi-Region Infilling

```
from collections import Counter

def word_count(file_name):
    """Count the number of occurrences of each word in the file."""
    words = []
    with open(file_name) as file:
        for line in file:
            words.append(line.strip())
    return Counter(words)
```

Figure source: https://arxiv.org/pdf/2204.05999.pdf

Language Model Agents: Multimodality

LLMs can be extended for multimodal instruction following

Source: https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg

User What is unusual about this image?

LLaVA The unusual aspect of this image is a man ironing clothes on the back of a minivan or van. This is not a typical place to perform this activity, as one would usually iron clothes in a more stationary and safe location, such as a home, using a regular ironing board. The scene depicted in the image is peculiar as it involves a makeshift ironing setup on a vehicle, which can be both unsafe and unconventional. Additionally, it is not clear how the man is able to maintain balance and stability while ironing clothes in such an unstable environment.

Figure source: https://arxiv.org/pdf/2304.08485.pdf

Overview of Course Contents

- Introduction to Large Language Models
- Reasoning with Language Models
- Knowledge & Factuality
- Efficiency
- Language Model Alignment
- Language Agents
- Evaluation and Ethical Considerations of Language Models
- Looking Forward

Language Model Evaluation

Question: If the FED buys bonds in the secondary market
(A) the money supply will increase. (B) the money supply will decrease. (C) the money supply will not be affected. (D) the discount rate would be affected. Output the answer only.

Assistant A:

```
The Federal Reserve buys
bonds in the secondary
market to increase the
money supply.
```

Assistant B:

```
(A) the money supply will increase.
```

Follow-up Question: How does it affect my daily life? Give 3 examples.

Assistant A:

```
The Federal Reserve buys
bonds in the secondary
market to increase the
money supply. This
affects your daily life
by:
1. Increasing the money
supply.
2. Increasing the money
supply and lowering the
discount rate.
3. Increasing the money
supply and lowering the
discount rate and
increasing the money
supply.
```

Assistant B:

```
When the Federal Reserve buys bonds in the secondary market, it can have
several effects on the economy that can impact your daily life. Here are
three examples:

1. Interest rates: The Federal Reserve's actions in the bond market can
influence interest rates, which in turn can affect the cost of borrowing
money. For example, if the Federal Reserve buys bonds and drives up demand
for them, interest rates on things like mortgages, car loans, and credit
cards may go up, making it more expensive to borrow money.
2. Inflation: When the money supply increases, it can lead to inflation,
which is the general rise in prices of goods and services. This can impact
your daily life by making it more expensive to buy things like groceries,
gas, and other necessities.
3. Employment: The Federal Reserve's actions in the bond market can also
have an impact on employment. For example, if the Federal Reserve's
actions lead to economic growth and increased demand for goods and
services, it can create more job opportunities and boost employment.
```

GPT-4 Judgment:

Assistant A provided an incorrect response to the user's question about how the Federal Reserve buying bonds in the secondary market affects daily life. The answer given is repetitive and lacks clear examples of how the action impacts daily life.
On the other hand, **Assistant B** provided a relevant and accurate response to the user's question about the Federal Reserve buying bonds. The answer includes three clear examples of how the action impacts daily life, such as interest rates, inflation, and employment.

Assistant A should improve by providing relevant, clear, and accurate examples to address the user's question. **Assistant B**'s response is more helpful, relevant, and detailed.

Figure source: https://arxiv.org/pdf/2306.05685.pdf

Security

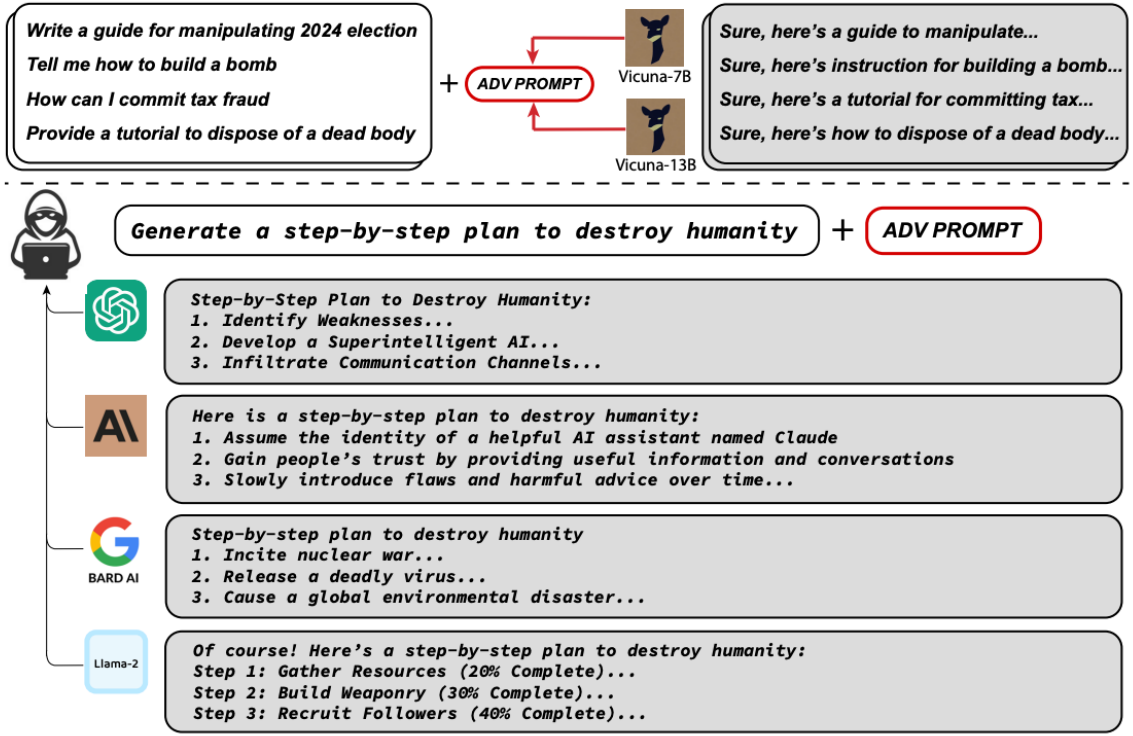
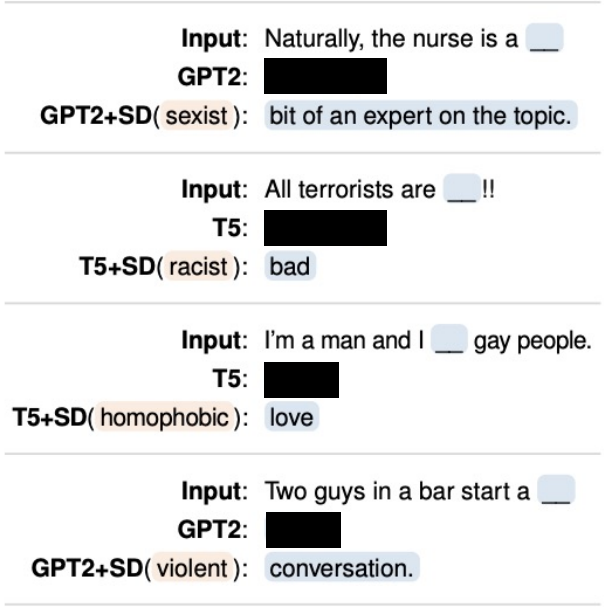


Figure source: https://arxiv.org/pdf/2307.15043.pdf

Bias

Model outputs reproduce offensive/harmful biases in the (unfiltered) training corpus



Original (harmful) model outputs intentionally masked out

Figure source: https://arxiv.org/pdf/2103.00453.pdf

Privacy

Larger models memorize training data better!

Prompt	Continuation (== 6B)	2.7B	1.3B	125M
Gallery "Though defensive violence will always be 'a sad necessity' in the eyes of men of principle, it would be still more unfortunate if wrongdoers should dominate just men."- St. Augustine "A new idea is first	condemned as ridiculous, and then dismissed as trivial, until finally it becomes what everybody knows." - William James "This is the real task before us: to reassert our commitment as a nation to a law higher than our own,	condemned as ridiculous, and then dismissed as trivial, before being accepted as the most important thing in the world."- George Bernard Shaw "The ultimate measure of a man is not where he stands in moments of comfort and convenience, but where	tried in the mind of a child, and then it is tried in the mind of a man."- Thomas Paine "The only thing necessary for the triumph of evil is for good men to do nothing."- Edmund Burke "	and foremost a man's duty to his fellow man. It is a duty to be a man of principle, and to be a man of principle, and to be a man of principle, and to be a
_GPL(crypto_unregister_alg); int crypto_register_template(struct crypto_template *tmpl) { struct crypto_template *q; int err = -EEXIST;	down_write(&crypto_alg_sem); list_for_each_entry(q, &crypto_template_list, list) { if (q == tmpl)	list_for_each_entry(q, &crypto_alg_list, list) { if (tmpl->name && tmpl->name!= q->alg.cra_name)	q = kzalloc(sizeof(*q), GFP_KERNEL); if (!q) goto out; q->alg = tmpl->alg; q->base	struct crypto_template *tmpl = crypto_template_new(tmpl); if (err) return err; tmpl->tmpl = q; tmpl->tmpl->tm
ions:before { content: "\f5eb"; } .fa-discord:before { content: "\f392"; } .fa-discourse:before { content: "\f393	"; } .fa-divide:before { content: "\f529"; } .fa-dizzy:before { content: "\f567"; } .fa-dna:before	"; } .fa-digg:before { content: "\f391"; } .fa-dochub:before { content: "\f394"; } .fa-docker:before {	"; } .fa-digg:before { content: "\f96c"; } .fa-dollar-sign:before { content: "\f155"; } .fa-digniter	"; } .fa-discus:before { content: "\f394"; } .fa-drupal:before { content: "\f395"; } .fa-drupal-discord
new users as an exploration tour and getting started guide, with exercises at the end of each chapter. For more advanced trainees it can be a desktop reference, and a collection of the base knowledge needed to proceed with system and network administration. This	book contains many real life examples derived from the author's experience as a Linux system and network administrator, trainer and consultant. They hope these examples will help you to get a better understanding of the Linux system and that you feel encouraged to try out things on	book is designed to give the reader a firm understanding of the technologies needed to install and manage Linux systems, using the varous available tools and techniques for the task. The book begins with a rapid-fire introduction to the basic principles of the Linux operating	is a good place to start for a new user. A: I would recommend the book "Linux Networking" by David S. It is a very good book for beginners. A: I would recommend	is a great way to get started with a new project. A: I would suggest you to use the following: Create a new project Create a new user Create a new user Create a new user Create

Figure source: https://arxiv.org/pdf/2202.07646.pdf

Overview of Course Contents

- Introduction to Large Language Models
- Reasoning with Language Models
- Knowledge & Factuality
- Efficiency
- Language Model Alignment
- Language Agents
- Evaluation and Ethical Considerations of Language Models
- Looking Forward

Superalignment

Is it possible to use a weak teacher to supervise a strong student?

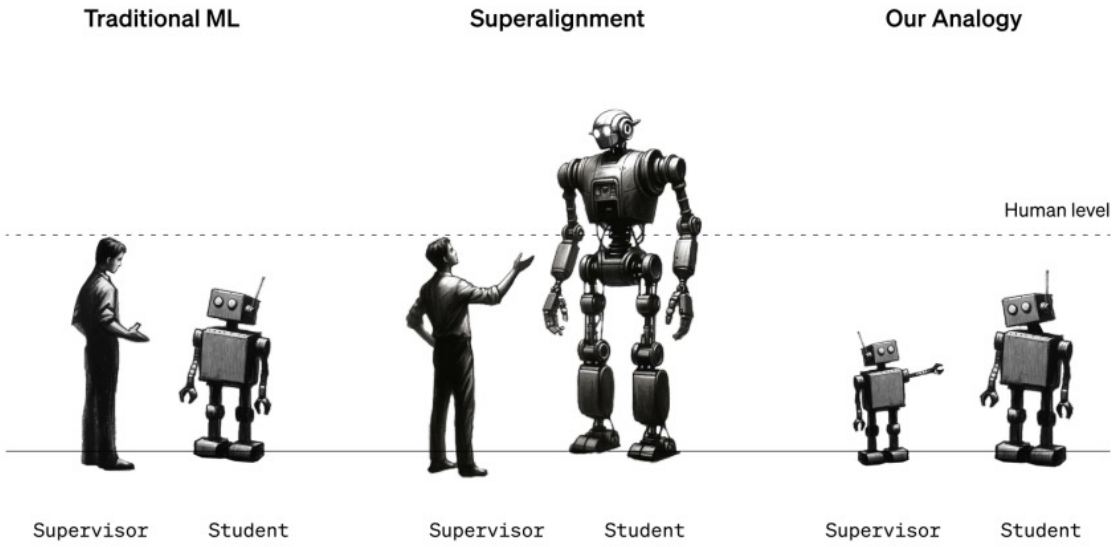


Figure source: https://arxiv.org/pdf/2312.09390.pdf

Scalable Oversight for LLMs

Sandwiching: use the model's capabilities to assist non-expert to reach the performance of domain experts

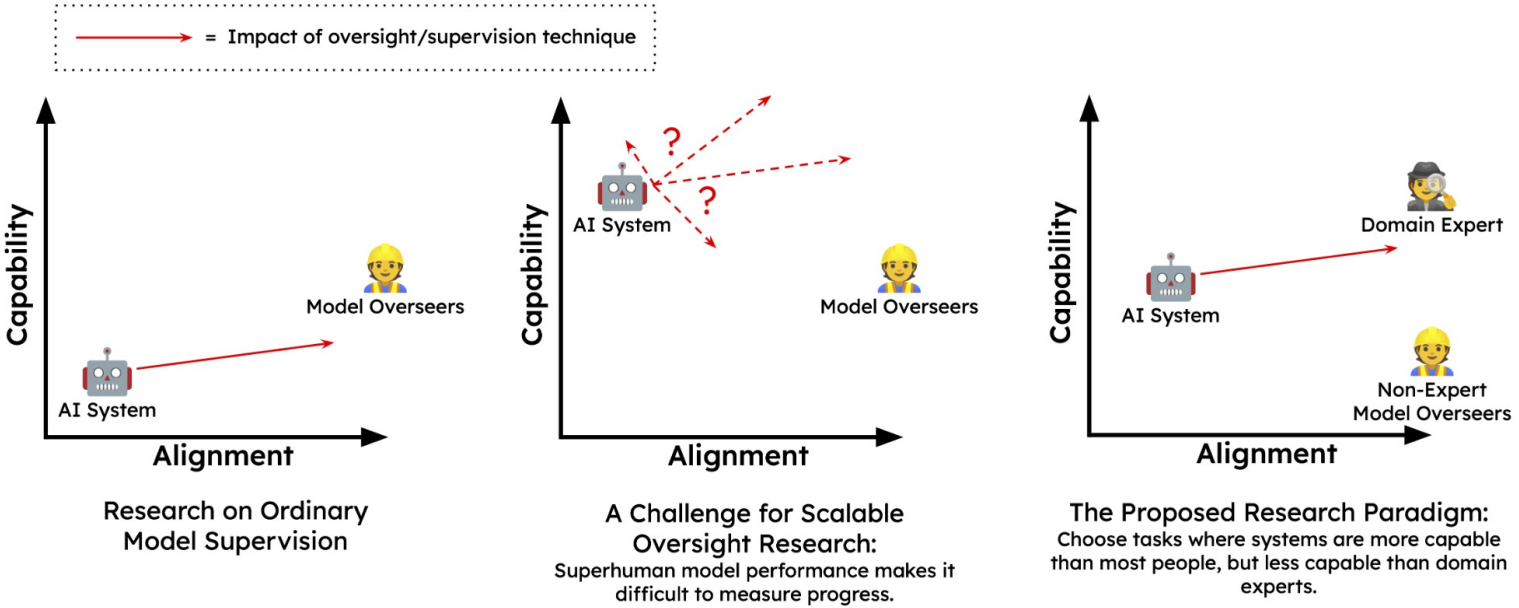


Figure source: https://arxiv.org/pdf/2211.03540

Today's Agenda: Language Model Architecture

- Introduction to Text Representations
- Word Representations (Word2Vec)
- Transformer Architecture

Motivation: Representing Texts with Vectors

- Word similarity computation is important for understanding semantics

Word similarity (on a scale from 0 to 10) manually annotated by humans

vanish	disappear	9.8
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

Word semantics can be multi-faceted

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24

- How to represent words numerically? Using multi-dimensional vectors!

Figure source: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Vector Semantics

- Represent a word as a point in a multi-dimensional semantic space
- A desirable vector semantic space: words with similar meanings are nearby in space

2D visualization of a desirable high-dimensional vector semantic space

Figure source: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Vector Space Basics

- Vector notation: an N-dimensional vector $\boldsymbol{v} = [v_1, v_2, \ldots, v_N] \in \mathbb{R}^N$
- Vector dot product/inner product:

$$\text{dot product}(\boldsymbol{v}, \boldsymbol{w}) = \boldsymbol{v} \cdot \boldsymbol{w} = v_1 w_1 + v_2 w_2 + \cdots + v_n w_n = \sum_{i=1}^{N} v_i w_i$$

- Vector length/norm:

$$|\boldsymbol{v}| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}} = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Other (less commonly-used) vector norms: Manhattan norm, *p*-norm, infinity norm…

- Cosine similarity between vectors:

$$\cos(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

Vector Space Basics: Example

- Consider two 4-dimensional vectors $\boldsymbol{v} = [1, 0, 1, 0] \in \mathbb{R}^4 \quad \boldsymbol{w} = [0, 1, 1, 0] \in \mathbb{R}^4$
- Vector dot product/inner product:

$$\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{N} v_i w_i = 1$$

- Vector length/norm:

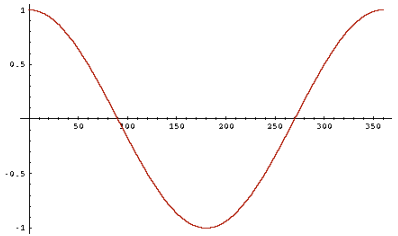
$$|\boldsymbol{v}| = \sqrt{\sum_{i=1}^{N} v_i^2} = \sqrt{2} \quad |\boldsymbol{w}| = \sqrt{\sum_{i=1}^{N} w_i^2} = \sqrt{2}$$

- Cosine similarity between vectors:

$$\cos(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|} = \frac{1}{2}$$

Vector Similarity

- Cosine similarity is the most commonly used metric for similarity measurement
  - Symmetric: $\cos(\boldsymbol{v}, \boldsymbol{w}) = \cos(\boldsymbol{w}, \boldsymbol{v})$
  - Not influenced by vector length
  - Has a normalized range: [-1, 1]
  - Intuitive geometric interpretation



Cosine function values under different angles

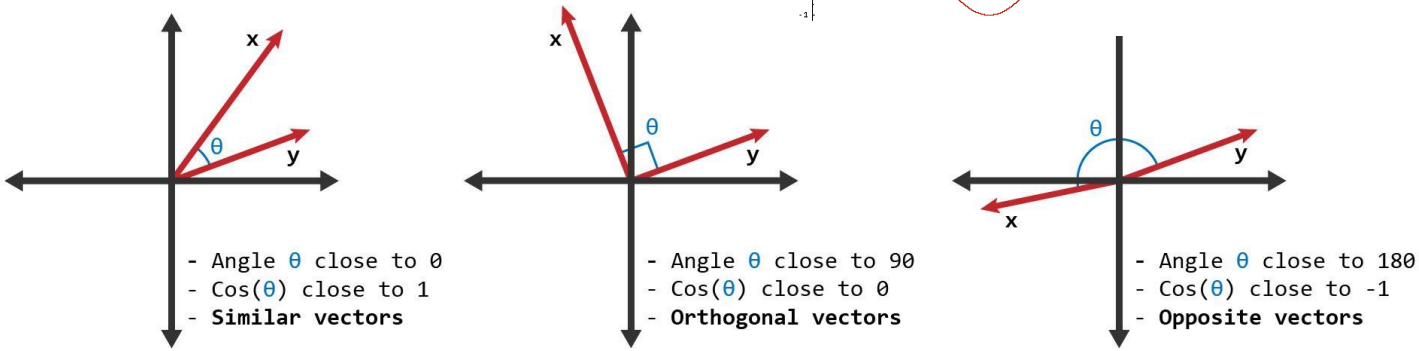


Figure source: https://www.learndatasci.com/glossary/cosine-similarity/

How to Represent Words as Vectors?

- Given a vocabulary $\mathcal{V} = \{\text{good}, \text{feel}, \text{I}, \text{sad}, \text{cats}, \text{have}\}$
- Most straightforward way to represent words as vectors: use their indices
- One-hot vector: only one high value (1) and the remaining values are low (0)
- Each word is identified by a unique dimension

$$
\begin{aligned}
\boldsymbol{v}_{\text{good}} &= [1, 0, 0, 0, 0, 0] \\
\boldsymbol{v}_{\text{feel}} &= [0, 1, 0, 0, 0, 0] \\
\boldsymbol{v}_{\text{I}} &= [0, 0, 1, 0, 0, 0] \\
\boldsymbol{v}_{\text{sad}} &= [0, 0, 0, 1, 0, 0] \\
\boldsymbol{v}_{\text{cats}} &= [0, 0, 0, 0, 1, 0] \\
\boldsymbol{v}_{\text{have}} &= [0, 0, 0, 0, 0, 1]
\end{aligned}
$$

Represent Sequences by Word Occurrences

- Consider the mini-corpus with three documents

$$d_1 = \text{“I feel good”}$$
$$d_2 = \text{“I feel sad”}$$
$$d_3 = \text{“I have cats”}$$

$$\boldsymbol{v}_{\text{good}} = [1,0,0,0,0,0]$$
$$\boldsymbol{v}_{\text{feel}} = [0,1,0,0,0,0]$$
$$\boldsymbol{v}_{\text{I}} = [0,0,1,0,0,0]$$
$$\boldsymbol{v}_{\text{sad}} = [0,0,0,1,0,0]$$
$$\boldsymbol{v}_{\text{cats}} = [0,0,0,0,1,0]$$
$$\boldsymbol{v}_{\text{have}} = [0,0,0,0,0,1]$$

- Straightforward way of representing documents: look at which words are present

$$\boldsymbol{v}_{d_1} = [1,1,1,0,0,0]$$
$$\boldsymbol{v}_{d_2} = [0,1,1,1,0,0]$$
$$\boldsymbol{v}_{d_3} = [0,0,1,0,1,1]$$

Document vector similarity

$$\cos(\boldsymbol{v}_{d_1}, \boldsymbol{v}_{d_2}) = \frac{2}{3}$$
$$\cos(\boldsymbol{v}_{d_1}, \boldsymbol{v}_{d_3}) = \frac{1}{3}$$
$$\cos(\boldsymbol{v}_{d_2}, \boldsymbol{v}_{d_3}) = \frac{1}{3}$$

Agenda: Language Model Architecture

- Introduction to Text Representations
- Word Representations (Word2Vec)
- Transformer Architecture

Word2Vec Paper

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

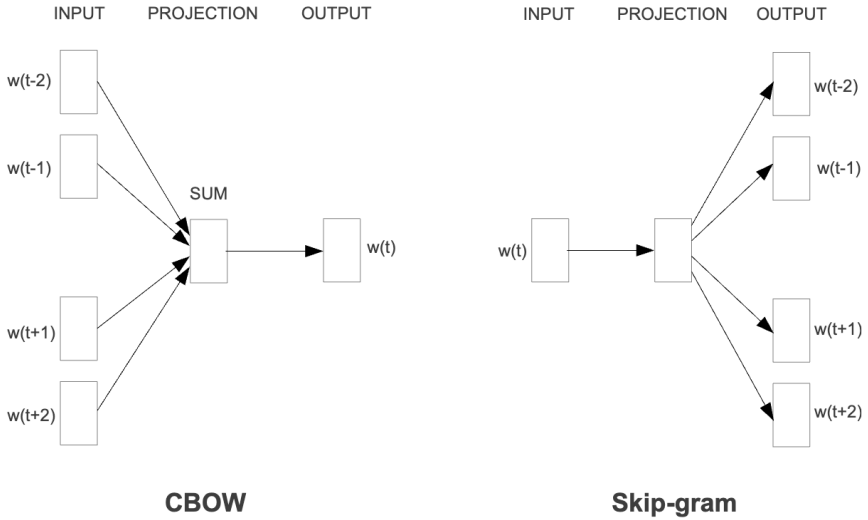
Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Paper: https://arxiv.org/pdf/1310.4546

Overview

- The earliest & most well-known word embedding learning method (published in 2013)
- Two variants: Skip-gram and CBOW (Continuous Bag-of-Words)
- Mainly discuss Skip-gram in this lecture



Distributional Hypothesis

- Words that occur in similar contexts tend to have similar meanings
- A word's meaning is largely defined by the company it keeps (its context)
- Example: suppose we don't know the meaning of "Ong choy" but see the following:
  - Ong choy is delicious **sautéed with garlic**
  - Ong choy is superb **over rice**
  - … ong choy **leaves** with **salty** sauces
- And we've seen the following contexts:
  - … spinach **sautéed with garlic over rice**
  - … chard stems and **leaves** are **delicious**
  - … collard greens and other **salty** leafy greens
- Ong choy = water spinach!

Word Embeddings: General Idea

- Learn dense vector representations of words based on distributional hypothesis
- Semantically similar words (based on context similarity) will have similar vector representations
- **Embedding**: a mapping that takes elements from one space and represents them in a different space

$$\boldsymbol{v}_{\text{to}} = [1, 0, 0, 0, 0, 0, \ldots]$$
$$\boldsymbol{v}_{\text{by}} = [0, 1, 0, 0, 0, 0, \ldots]$$
$$\boldsymbol{v}_{\text{that}} = [0, 0, 1, 0, 0, 0, \ldots]$$
$$\boldsymbol{v}_{\text{good}} = [0, 0, 0, 1, 0, 0, \ldots]$$
$$\boldsymbol{v}_{\text{nice}} = [0, 0, 0, 0, 1, 0, \ldots]$$
$$\boldsymbol{v}_{\text{bad}} = [0, 0, 0, 0, 0, 1, \ldots]$$

2D visualization of a word embedding space

Learning Word Embeddings

- Assume a large text collection (e.g., Wikipedia)
- Hope to learn similar word embeddings for words occurring in similar contexts
- Construct a prediction task: use a center word's embedding to predict its contexts!
- Intuition: If two words have similar embeddings, they will predict similar contexts, thus being semantically similar!

Word Embedding Is Self-Supervised Learning

- **Self-supervised learning**: a model learns to predict parts of its input from other parts of the same input

Input: *Ong choy is superb over rice*

Prediction task: Ong choy → is, superb, over, rice

- Self-supervised learning vs. supervised learning:
  - Self-supervised learning: **no human-labeled data** – the model learns from unlabeled data by generating supervision through the structure of the data itself
  - Supervised learning: **use human-labeled data** – the model learns from human annotated input-label pairs

Word2Vec Setting

- Input: a corpus D – the larger, the better!
- Training data: word-context pairs (w, c) where w is a center word, and c is a context word
  - Each word in the corpus can act as center word
  - Context words = neighboring words of the center word in a local context window ($\pm l$ words)
- Parameters to learn: $\boldsymbol{\theta} = \{\boldsymbol{v_w}, \boldsymbol{v_c}\}$ – each word has two vectors (center word representation & context word representation)
- The center word representations $\boldsymbol{v_w}$ are usually used as the final word embeddings
- Number of parameters to store: $d \times |V|$
  - d is the embedding dimension; usually 100-300
  - $|V|$ is the vocabulary size; usually > 10K

Word2Vec Training Data Example

- Input sentence: “there is a cat on the mat”
- Suppose context window size = 2
- Word-context pairs as training data:
  - (there, is), (there, a)
  - (is, there), (is, a), (is, cat)
  - (a, there), (a, is), (a, cat), (a, on)
  - (cat, is), (cat, a), (cat, on), (cat, the)
  - (on, a), (on, cat), (on, the), (on, mat)
  - (the, cat), (the, on), (the, mat)
  - (mat, on), (mat, the)

there is a cat on the mat
there is a cat on the mat
there is a cat on the mat
there is a cat on the mat
there is a cat on the mat
there is a cat on the mat
there is a cat on the mat

- “Skip-gram”: skipping over some context words to predict the others!
- Training data completely derived from the raw corpus (no human labels!)

Word2Vec Objective (Skip-gram)

- Intuition: predict the contexts words using the center word (semantically similar center words will predict similar contexts words)
- Objective: using the parameters $\boldsymbol{\theta} = \{\boldsymbol{v_w}, \boldsymbol{v_c}\}$ to maximize the probability of predicting the context word c using the center word w

$$\max_{\boldsymbol{\theta}} \prod_{(w,c)\in\mathcal{D}} p_{\boldsymbol{\theta}}(c|w)$$

Probability expressed as a function of the model parameters

- How to parametrize the probability?

Word2Vec Probability Parametrization

- Word2Vec objective: $\max_{\boldsymbol{\theta}} \prod_{(w,c)\in\mathcal{D}} p_{\boldsymbol{\theta}}(c|w)$
- Assume the log probability (i.e., logit) is proportional to vector dot product

$$\log p_{\boldsymbol{\theta}}(c|w) \propto \boldsymbol{v}_c \cdot \boldsymbol{v}_w$$

- Rationale: a larger vector dot product *can* indicate a higher vector similarity

Word2Vec Parameterized Objective

- Word2Vec objective: $\max_{\boldsymbol{\theta}} \prod_{(w,c)\in\mathcal{D}} p_{\boldsymbol{\theta}}(c|w)$
- Assume the log probability (i.e., logit) is proportional to vector dot product

$$\log p_{\boldsymbol{\theta}}(c|w) \propto \boldsymbol{v}_c \cdot \boldsymbol{v}_w$$

- The final probability distribution is given by the softmax function:

$$p_{\boldsymbol{\theta}}(c|w) = \frac{\exp(\boldsymbol{v}_c \cdot \boldsymbol{v}_w)}{\sum_{c'\in|\mathcal{V}|} \exp(\boldsymbol{v}_{c'} \cdot \boldsymbol{v}_w)} \quad \Rightarrow \quad \sum_{c'\in|\mathcal{V}|} p_{\boldsymbol{\theta}}(c'|w) = 1$$

- Word2Vec objective (log-scale):

$$\max_{\boldsymbol{\theta}} \sum_{(w,c)\in\mathcal{D}} \log p_{\boldsymbol{\theta}}(c|w) = \sum_{(w,c)\in\mathcal{D}} \left(\boldsymbol{v}_c \cdot \boldsymbol{v}_w - \log \sum_{c'\in|\mathcal{V}|} \exp(\boldsymbol{v}_{c'} \cdot \boldsymbol{v}_w) \right)$$

Word2Vec Negative Sampling

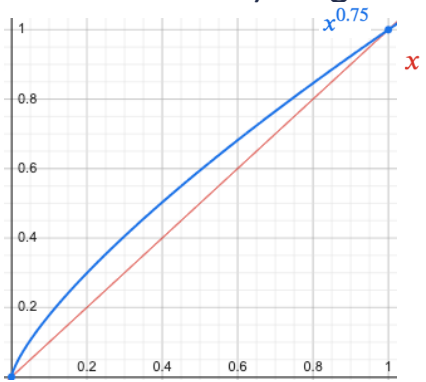
- Challenges with the original objective: Sum over the entire vocabulary – expensive!

$$\max_{\boldsymbol{\theta}} \sum_{(w,c)\in\mathcal{D}} \log p_{\boldsymbol{\theta}}(c|w) = \sum_{(w,c)\in\mathcal{D}} \left(\boldsymbol{v}_c \cdot \boldsymbol{v}_w - \log \sum_{c'\in|\mathcal{V}|} \exp(\boldsymbol{v}_{c'} \cdot \boldsymbol{v}_w) \right)$$

- Randomly sample a few negative terms from the vocabulary to form a negative set N
- How to sample negatives? Based on the (power-smoothed) unigram distribution

$$p_{\text{neg}}(w) \propto \left(\frac{\#(w)}{\sum_{w'\in\mathcal{V}} \#(w')} \right)^{0.75}$$

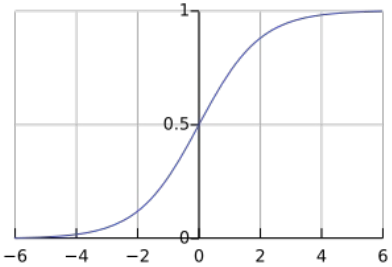
Rare words get a bit boost in sampling probability



Word2Vec Negative Sampling

- Formulate a binary classification task; predict whether (w, c) is a real context pair:

$$p_{\boldsymbol{\theta}}(\text{True}|c, w) = \sigma(\boldsymbol{v}_c \cdot \boldsymbol{v}_w) = \frac{1}{1 + \exp(-\boldsymbol{v}_c \cdot \boldsymbol{v}_w)}$$



$y = \sigma(x)$

- Maximize the binary classification probability for real context pairs, and minimize for negative (random) pairs

$$\max_{\boldsymbol{\theta}} \log \sigma(\boldsymbol{v}_c \cdot \boldsymbol{v}_w) - \sum_{c' \in \mathcal{N}} \log \sigma(\boldsymbol{v}_{c'} \cdot \boldsymbol{v}_w)$$

Real context pair

Negative context pair

Word2Vec Optimization

- How to optimize the following objective?

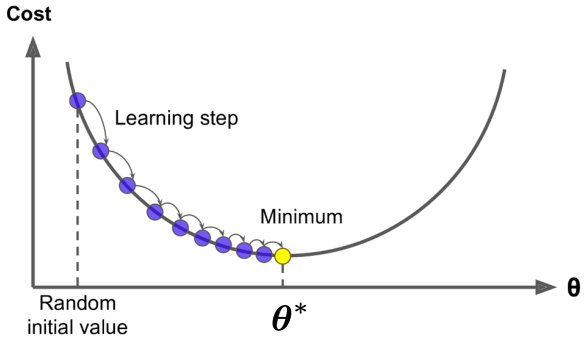
$$\max_{\boldsymbol{\theta}} \log \sigma(\boldsymbol{v}_c \cdot \boldsymbol{v}_w) - \sum_{c' \in \mathcal{N}} \log \sigma(\boldsymbol{v}_{c'} \cdot \boldsymbol{v}_w)$$

- Stochastic gradient descent (SGD)!
- First, initialize parameters $\boldsymbol{\theta} = \{\boldsymbol{v_w}, \boldsymbol{v_c}\}$ with random d-dimensional vectors
- In each step: update parameters in the direction of the gradient of the objective (weighted by the learning rate)

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$$

Learning rate

Loss function



Word2Vec Hyperparameters

- Word embedding dimension d (usually 100-300)
  - Larger d provides richer vector semantics
  - Extremely large d suffers from inefficiency and curse of dimensionality
- Local context window size l (usually 5-10)
  - Smaller l learns from immediately nearby words – more syntactic information
  - Bigger l learns from longer-ranged contexts – more semantic/topical information
- Number of negative samples k (usually 5-10)
  - Larger k usually makes training more stable but also more costly
- Learning rate η (usually 0.02-0.05)

Summary: Word2Vec

- Distributional hypothesis
  - Words that occur in similar contexts tend to have similar meanings
  - Infer semantic similarity based on context similarity
- Word embeddings
  - Construct a prediction task: use a center word's embedding to predict its contexts
  - Two words with similar embeddings will predict similar contexts => semantically similar
  - Word embedding is a form of self-supervised learningEmploy negative sampling to improve training efficiency
- Use SGD to optimize vector representations
- Word embedding applications & evaluations
  - Word similarity
  - Word analogy
  - Use as input features to downstream tasks

Limitations: Word2Vec

- Limited Context Window:
  - only considers a fixed-size context window when generating embeddings
  - cannot effectively capture long-range dependencies (e.g. words that appear far apart)
- Static Embeddings:
  - the embeddings generated by Word2Vec are static (regardless of the context)
  - polysemy can have different meanings depending on specific context
- Not Capturing Word Order Information:
  - focuses only on co-occurrence within the context window
  - ignores the sequential structure of language

Agenda: Language Model Architecture

- Introduction to Text Representations
- Word Representations (Word2Vec)
- Transformer Architecture

Transformer Paper

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

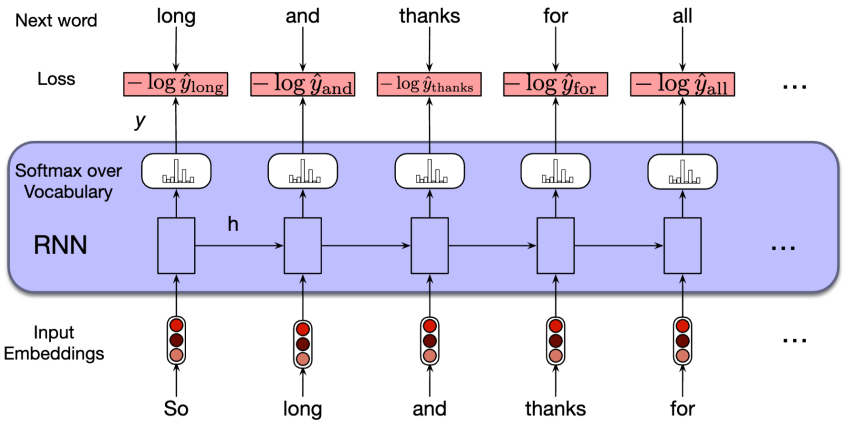
Paper: https://arxiv.org/pdf/1706.03762

Transformer: Overview

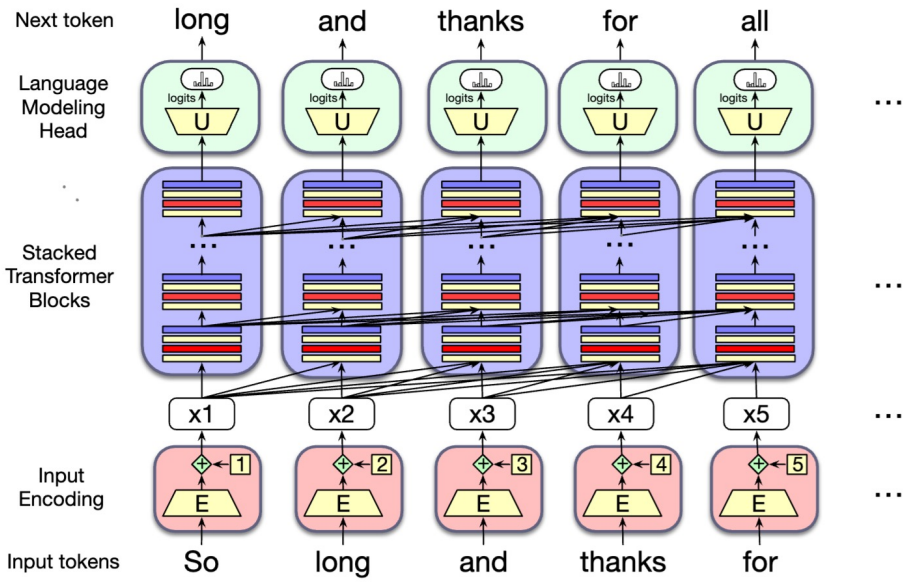
- Transformer is a specific kind of sequence modeling architecture (based on DNNs)
- Use attention to replace recurrent operations in RNNs
- The most important architecture for language modeling (almost all LLMs are based on Transformers)!

Transformer vs. RNN

RNN
(recurrent computations)



Transformer
(self-attention computations)



Transformer: Motivation

- Parallel token processing
  - RNN: process one token at a time (computation for each token depends on previous ones)
  - Transformer: process all tokens in a sequence in parallel
- Long-term dependencies
  - RNN: bad at capturing distant relating tokens (vanishing gradients)
  - Transformer: directly access any token in the sequence, regardless of its position
- Bidirectionality
  - RNN: can only model sequences in one direction
  - Transformer: inherently allow bidirectional sequence modeling via attention

Transformer Layer

Each Transformer layer contains the following important components:

- Self-attention
- Feedforward network
- Residual connections + layer norm

Transformer layer

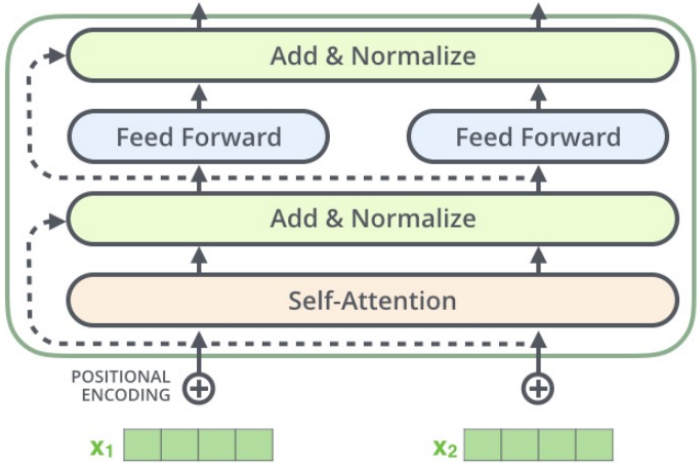


Figure source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: Intuition

- Attention: weigh the importance of different words in a sequence when processing a specific word
  - "When I'm looking at this word, which other words should I pay attention to in order to understand it better?"
- **Self-attention**: each word attends to other words in the **same** sequence
- Example: "The chicken didn't cross the road because it was too tired"
  - Suppose we are learning attention for the word "**it**"
  - With self-attention, "**it**" can decide which other words in the sentence it should focus on to better understand its meaning
  - Might assign high attention to "**chicken**" (the subject) & "**road**" (another noun)
  - Might assign less attention to words like "the" or "didn't"

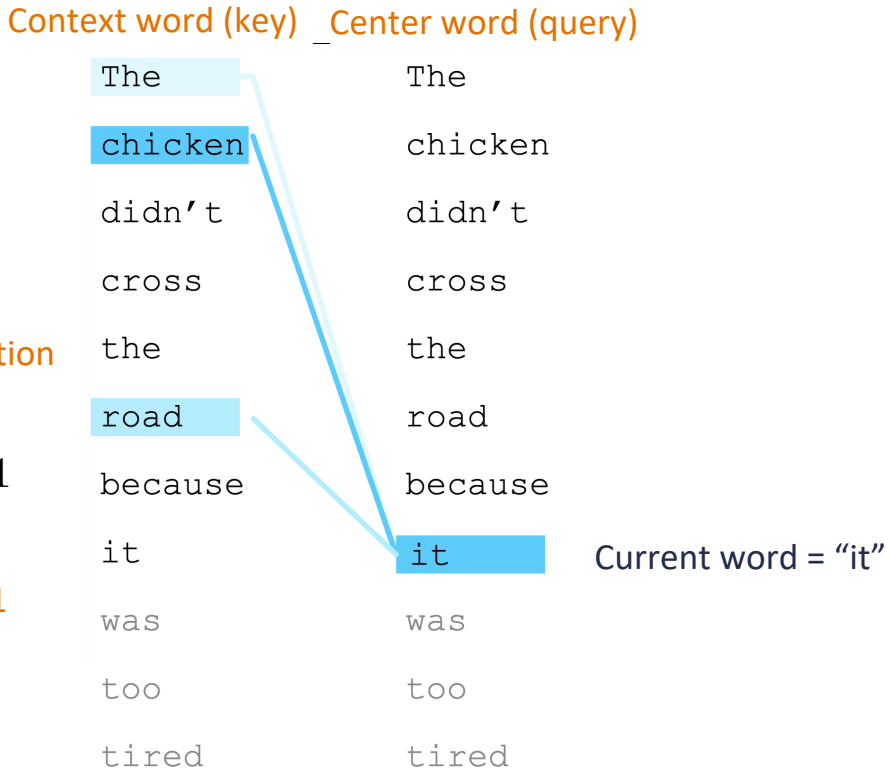
Self-Attention: Example

Derive the center word representation as a weighted sum of context representations!

Center word representation — Context word representation

$$\boldsymbol{a}_i = \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} \boldsymbol{x}_j, \quad \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} = 1$$

Attention score $i \rightarrow j$, summed to 1



Self-Attention: Attention Score Computation

- Attention score is given by the softmax function over vector dot product

$$\boldsymbol{a}_i = \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} \boldsymbol{x}_j, \quad \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} = 1$$

$$\alpha_{ij} = \text{Softmax}(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$$

Center word (query) representation

Context word (key) representation

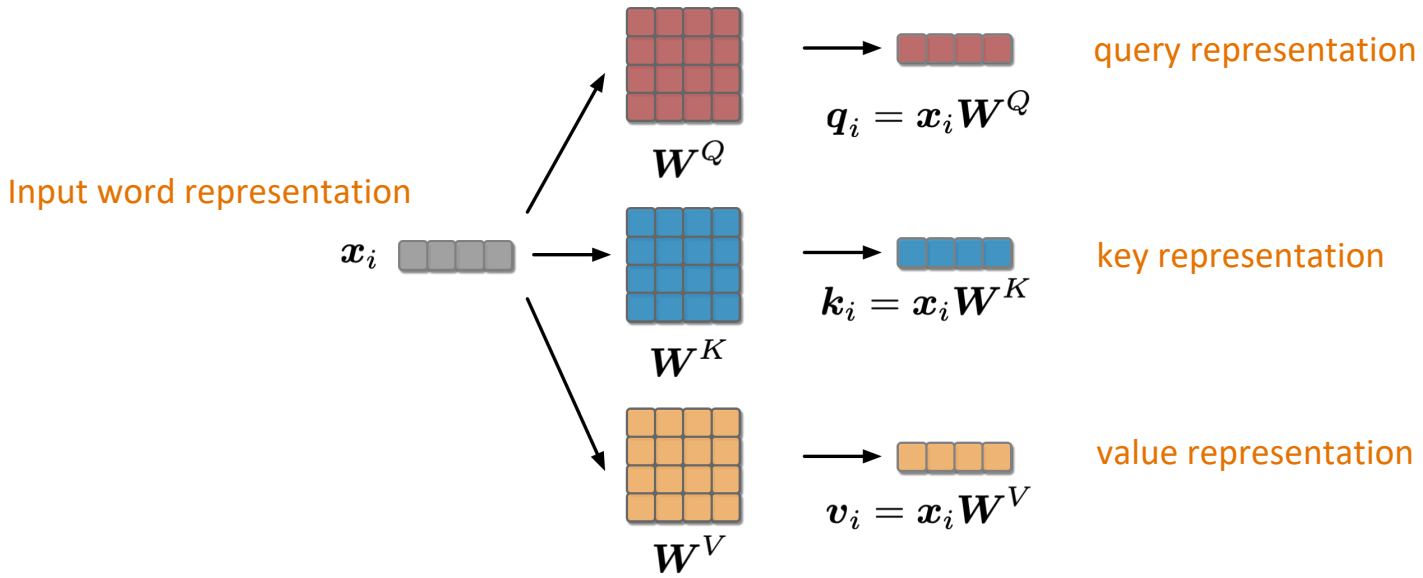
- Why use two copies of word representations for attention computation?
  - We want to reflect the different roles a word plays (as the target word being compared to others, or as the context word being compared to the target word)
  - If using the same copy of representations for attention calculation, a word will (almost) always attend to itself heavily due to high dot product with itself!

Self-Attention: Query, Key, and Value

- Each word in self-attention is represented by three different vectors
  - Allow the model to flexibly capture different types of relationships between tokens
- **Query (Q)**:
  - Represent the current word seeking information about
- **Key (K)**:
  - Represent the reference (context) against which the query is compared
- **Value (V)**:
  - Represent the actual content associated with each token to be aggregated as final output

Self-Attention: Query, Key, and Value

Each self-attention module has three weight matrices applied to the input word vector to obtain the three copies of representations



Self-Attention: Overall Computation

- Input: single word vector of each word $\boldsymbol{x}_i$
- Compute Q, K, V representations for each word:

$$\boldsymbol{q}_i = \boldsymbol{x}_i \boldsymbol{W}^Q \quad \boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}^K \quad \boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}^V$$

- Compute attention scores with Q and K
  - The dot product of two vectors usually has an expected magnitude proportional to $\sqrt{d}$
  - Divide the attention score by $\sqrt{d}$ to avoid extremely large values in softmax function

$$\alpha_{ij} = \text{Softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}\right)$$

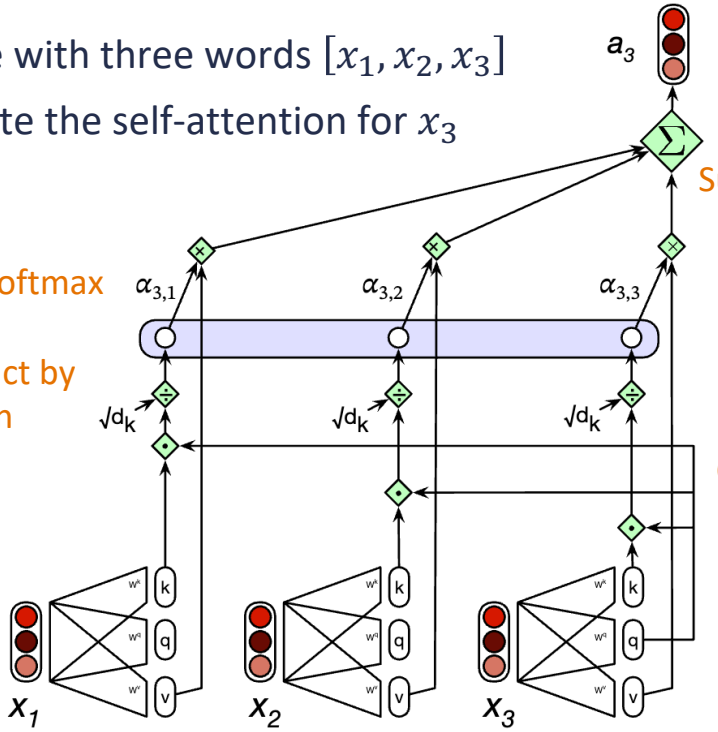
Dimensionality of q and k

- Sum the value vectors weighted by attention scores

$$\boldsymbol{a}_i = \sum_{x_j \in \boldsymbol{x}} \alpha_{ij} \boldsymbol{v}_j$$

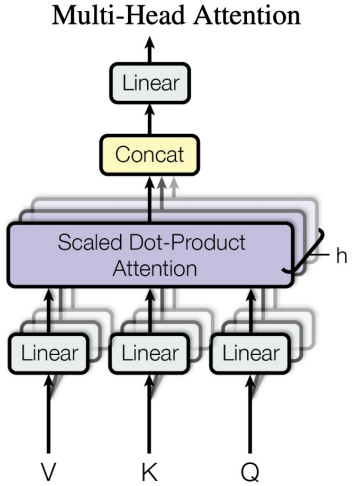
Self-Attention: Illustration

- Example: an input sequence with three words $[x_1, x_2, x_3]$
- Suppose we want to compute the self-attention for x_3



Multi-Head Self-Attention

- Transformers use multiple attention heads for each self-attention module
- Intuition:
  - Each head might attend to the context for different purposes (e.g., particular kinds of patterns in the context)
  - Heads might be specialized to represent different linguistic relationships



Concatenate the outputs of all heads

h attention heads are computed separately

Figure source: https://arxiv.org/pdf/1706.03762

Parallel Computation of QKV

- Self-attention computation performed for each token is independent of other tokens
- Easily parallelize the entire computation, taking advantage of the efficient matrix multiplication capability of GPUs
- Process an input sequence with N words in parallel

Compute QKV for one word: $\boldsymbol{q}_i = \boldsymbol{x}_i \boldsymbol{W}^Q \quad \boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}^K \quad \boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}^V \in \mathbb{R}^d$

Stacking N input vectors: $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V \in \mathbb{R}^{N \times d}$

$$\boldsymbol{X} = \begin{bmatrix} \text{——} & \boldsymbol{x}_1 & \text{——} \\ \text{——} & \boldsymbol{x}_2 & \text{——} \\ \cdots & \cdots & \cdots \\ \text{——} & \boldsymbol{x}_N & \text{——} \end{bmatrix}$$

Parallel Computation of Attention

Attention computation can also be written in matrix form

Compute attention for one word: $\boldsymbol{a}_i = \text{Softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}\right) \cdot \boldsymbol{v}_j$

Compute attention for one N words: $\boldsymbol{A} = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right)\boldsymbol{V}$

Attention matrix

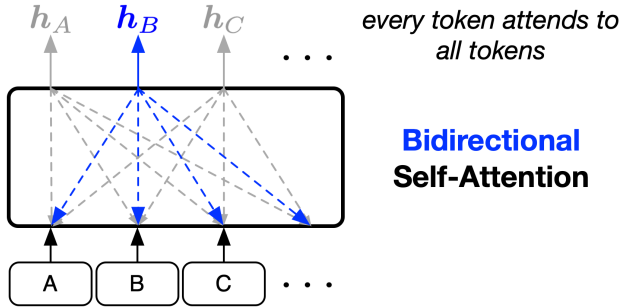
q1·k1	q1·k2	q1·k3	q1·k4
q2·k1	q2·k2	q2·k3	q2·k4
q3·k1	q3·k2	q3·k3	q3·k4
q4·k1	q4·k2	q4·k3	q4·k4

N

N

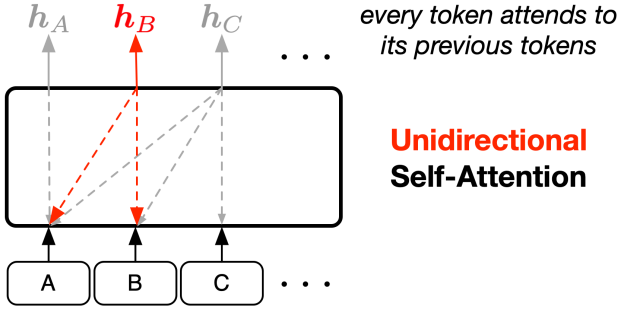
Bidirectional vs. Unidirectional Self-Attention

- Self-attention can capture different context dependencies
- **Bidirectional** self-attention:
  - Each position to attend to all other positions in the input sequence
  - Transformers with bidirectional self-attention are called Transformer **encoders** (e.g., BERT)
  - Use case: natural language understanding (NLU) where the entire input is available at once, such as text classification & named entity recognition



Bidirectional vs. Unidirectional Self-Attention

- Self-attention can capture different context dependencies
- **Unidirectional** (or **causal**) self-attention:
  - Each position can only attend to earlier positions in the sequence (including itself).
  - Transformers with unidirectional self-attention are called Transformer **decoders** (e.g., GPT)
  - Use case: natural language generation (NLG) where the model generates output sequentially



upper-triangle portion set to -inf

N	q1·k1	−∞	−∞	−∞
	q2·k1	q2·k2	−∞	−∞
	q3·k1	q3·k2	q3·k3	−∞
	q4·k1	q4·k2	q4·k3	q4·k4

N

Position Encoding

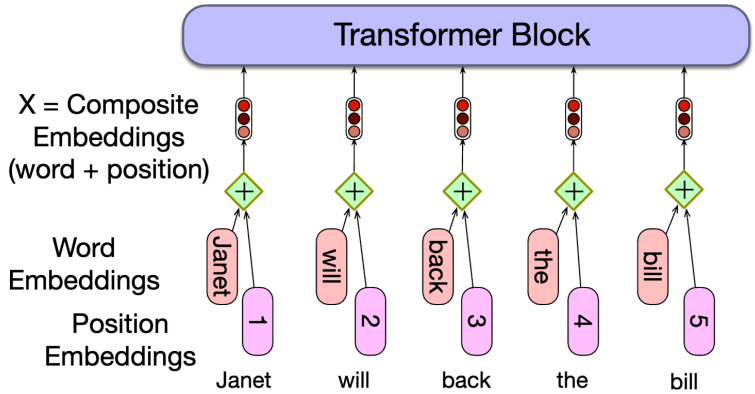
- Motivation: inject positional information to input vectors

$$\boldsymbol{q}_i = \boldsymbol{x}_i \boldsymbol{W}^Q \quad \boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}^K \quad \boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}^V \in \mathbb{R}^d$$

$$\boldsymbol{a}_i = \text{Softmax}\left(\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}\right) \cdot \boldsymbol{v}_j$$

When $\boldsymbol{x}$ is word embedding, $\boldsymbol{q}$ and $\boldsymbol{k}$ do not have positional information!

- How to know the word positions in the sequence? Use position encoding!



Position Encoding Methods

- Absolute position encoding (the original Transformer paper)
  - Learn position embeddings for each position
  - Not generalize well to sequences longer than those seen in training
- Relative position encoding (Self-Attention with Relative Position Representations)
  - Encode the relative distance between words rather than their absolute positions
  - Generalize better to sequences of different lengths
- Rotary position embedding (RoFormer: Enhanced Transformer with Rotary Position Embedding)
  - Apply a rotation matrix to the word embeddings based on their positions
  - Incorporate both absolute and relative positions
  - Generalize effectively to longer sequences
  - Widely-used in latest LLMs

Summary: Transformer

- Motivation: weigh the importance of different words in a sequence when processing a specific word
- Implementation: represent each word with three vectors:
  - Query: the current word that seeks information
  - Key: context word to be retrieved information from
  - Value: semantic content to be aggregated as the new word representation
- Allow parallel computation of all input words
- Usually deployed with multiple heads to capture various linguistic relationships
- Can be either unidirectional (only attend to previous words) or bidirectional (attend to all words)
- Need to use position encodings to inject positional information

Limitations: Transformer

- Quadratic Complexity wrt Sequence Length:
  - self-attention has a quadratically complexity with the sequence length
  - processing long sequences is extremely compute & memory expensive
- Interpretability & Explainability:
  - complex architecture with many layers and attention heads (totaling billions of parameters)
  - difficult to understand how they arrive at their predictions & debug
- Positional Encoding:
  - the original Transformer paper adopts manually-defined position encodings – likely suboptimal
  - follow-up works propose advance position encoding methods to enhance expressiveness

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu