



BITS Pilani
Pilani Campus

Conversational Systems Design

Kedar Kanhere
26/05/2024

Session Content

- Objective of course
 - Evaluation Plan
 - Introduction to Conversational AI
 - Introduction to conversational chatbots
 - Conversational Agents
 - Core technologies in conversational agents
 - Evolution of conversational agents and role of AI
 - Overview of chatbot deployment & flow based chatbots
-

Introduction

- This course offers a deep dive into the mechanics of chatbots and voice assistants, focusing on their design, functionality, and impact
- As digital assistants become more prevalent, understanding how to design effective conversational systems is crucial for the next generation of tech professionals.



Siri



Alexa



Cortana

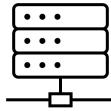


OK Google



Bixby

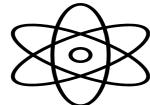
What Will You Learn



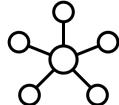
Foundations And
Advanced NLU



NLG And Dialogue
Systems



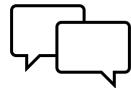
Deep Learning
in AI



Conversational
AI Architecture



AI Ethics and UX



ChatBot
Implementation



explainable AI &
context-aware
chatbots



AI Research and
Development



Optimization
Techniques



Capstone Project

Potential of Conversational AI

Career Opportunities: Skilled professionals are in high demand in fields like AI, machine learning, and user experience design.

Hands-On Experience: Gain practical experience with the tools and technologies that drive conversational AI, including NLP and multimodal interactions.

Innovative Learning: Participate in projects that mirror real-world applications and challenges, enhancing your problem-solving and technical skills.



Pedagogy and Evaluation Scheme

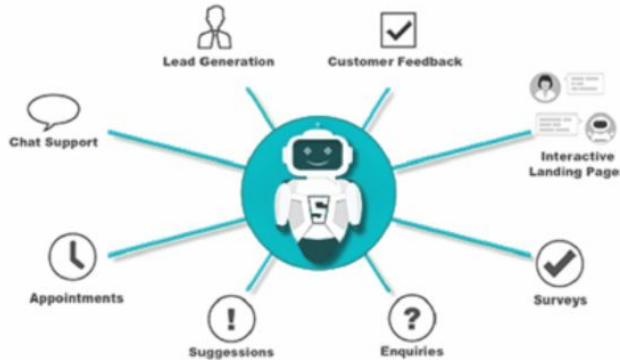
Pre-requisites:

- Basic Python
- Machine Learning Fundamentals
- Familiarity with JavaScript and Web APIs
- Deep learning
- Open Mind to learn new things

Exam Type	Date	Time	Total Marks (across exam type)
Quizzes (2 quizzes total)	1 quiz before mid sem and 1 after mid sem	NA	10 Marks
Mid Sem Exam	19th July	9 am to 11 am	20 Marks
Assignments (2 group assignments)	Both assignments post mid sem exam	NA	40 Marks
End Sem/Final Exam	27th September	9 am to 11:30 am	30 Marks

Introduction to Conversational AI

- What is Conversational AI?
 - Explain that Conversational AI involves technologies that enable computers to understand, process, and respond to human language in a natural way.
- Purpose:
 - To automate communication and create personalised customer experiences at scale.



Types of Conversational Agents

- Rule-Based vs. AI-Driven Systems:
 - **Rule-Based Systems:** Operate based on predefined responses.
 - **AI-Driven Systems:** Use machine learning to understand context and improve responses.
- Application Examples:
 - Customer support bots, virtual personal assistants, interactive voice response systems.



Core Tech in Conversational AI

- **Natural Language Processing (NLP):**
 - Techniques to understand human language, like tokenisation, stemming, and parsing.
- **Machine Learning:**
 - How AI learns from data to improve its language understanding and responses.
- **Key Components:**
 - Intent recognition, entity extraction, dialogue management.

Evolution of Conversational Systems

Initial Systems

- script-based
- limited to specific rules
- could not handle unexpected user inputs.

Example: Simple customer service FAQs where each question has a pre-defined response.

Development of Machine Learning Models

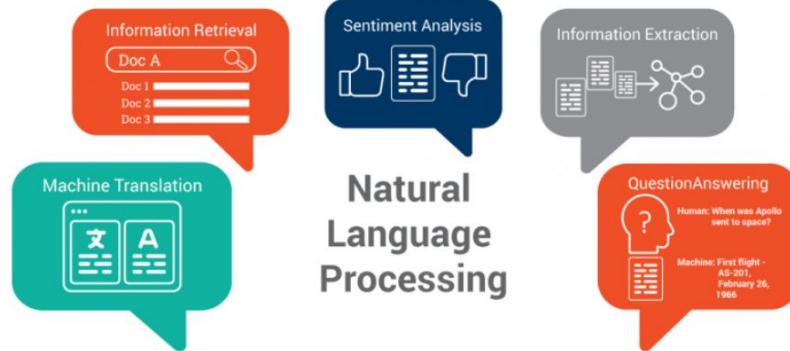
- systems learnt from vast datasets improving their ability to understand context and ambiguity in language.

Example: Chatbots in retail websites providing product recommendations based on customer interactions.

Evolution of Conversational Systems

Incorporation of NLP Advances

Virtual assistants like Google Assistant and Alexa, which understand and execute complex commands across various domains.



Evolution of Conversational Systems

Integration with Omni-channel Services

- Integration across multiple channels (web, mobile, voice)
- Maintaining context across sessions and platforms.

Example: Seamless customer support that continues conversations from text to voice calls without losing context.

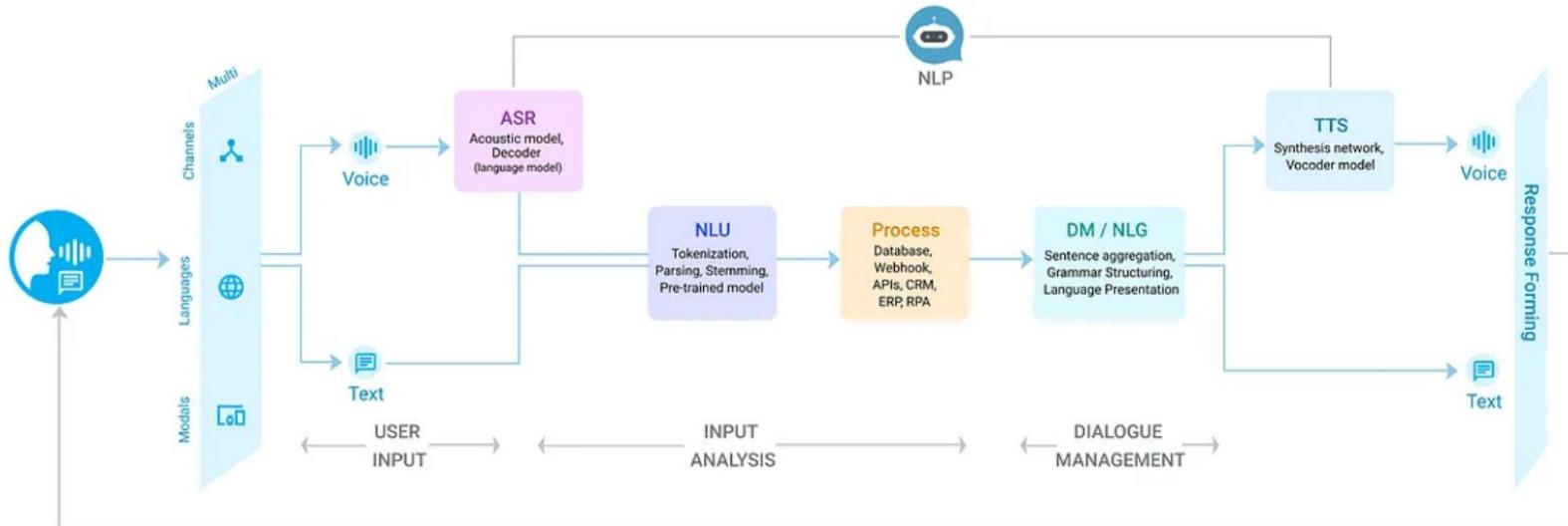
Impact on User Experience:

- Enhanced Accessibility
- Increased Personalisation
- Improved efficiency and satisfaction
- Consistency across channels

Role of AI Conversational Systems

- Adaptive Learning
- Enhancing Understanding and Interactions

Building a Conversational System



Building a Conversational System

Design Considerations:

- **Goals:** purpose (customer support, sales, etc.).
- **User Experience:** Plan the conversational flows to mimic natural dialogues.
- **Integration:** Ensure compatibility with existing technology platforms.

Development Tools and Platforms:

- **Tools:** Highlight tools like Google Dialogflow, IBM Watson, and Microsoft Bot Framework.
- **Capabilities:** platform specialises in (e.g., voice recognition, text processing).

Building a Conversational System

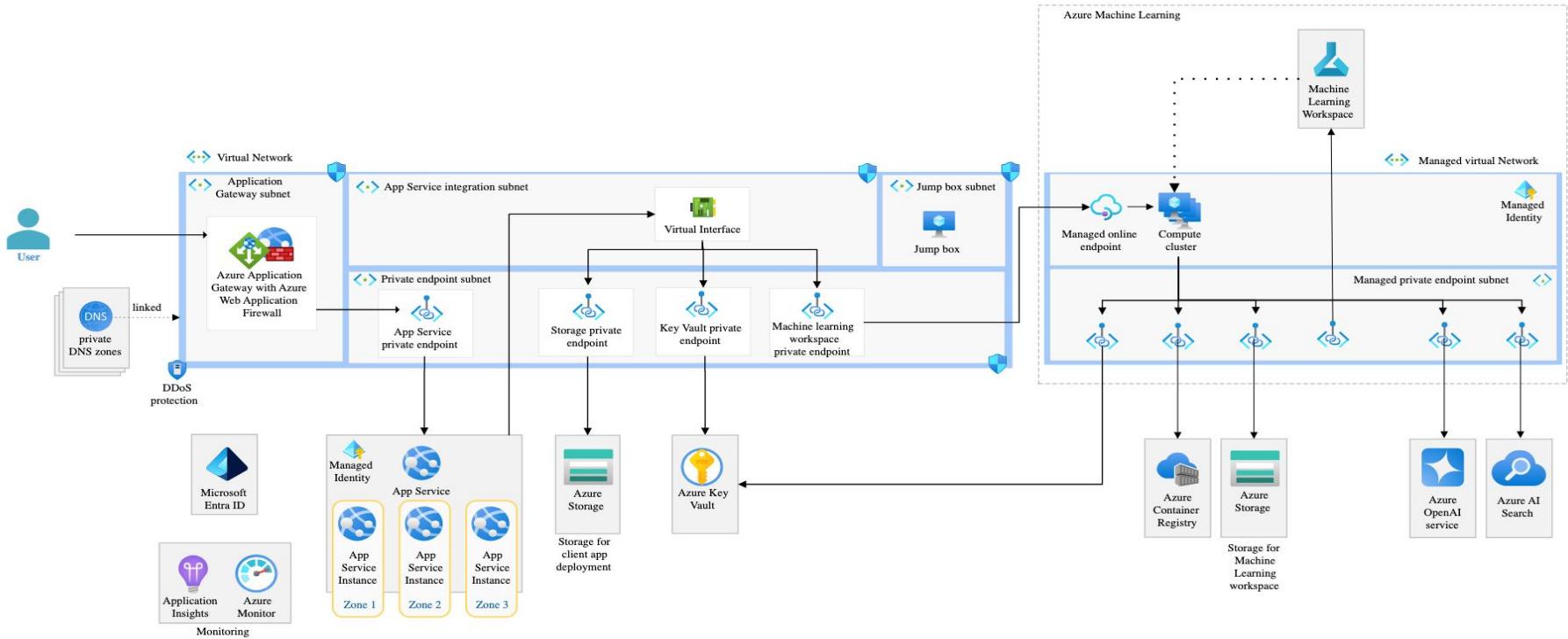
Implementation Steps:

- **Training:** Use relevant data to train the AI model on language and context.
- **Testing:** Continuously test and refine for understanding accuracy and user satisfaction.
- **Deployment:** Launch the system for real-world interaction, with ongoing monitoring.

Maintenance and Optimisation:

- **Updates:** Regular updates based on user feedback and evolving data.
- **Scaling:** Expand the system's capabilities as usage grows and needs change.

Deploying a Conversational System



Examples of a Conversational System

- Banks chatbots
- Website help chatbots which have flows in them
- Context based chatbots (Amazon)

Challenges / Future Direction

Challenges:

- Handling diverse languages, understanding nuances, maintaining privacy.

Future Trends:

- Predictions about conversational AI's role in more sectors and becoming more context-aware.

Q/A



Conversational Systems Design

Lecture 2



BITS Pilani
Pilani Campus

Kedar Kanhere
02/06/2024

Session Content but before that

Quiz and Assignment Tentative Schedule

Exam Type	Start Date	End Date	Marks
Quiz 1	13/07/2024 7:00 PM	14/07/2024 7:00 PM	5 (Best 2 of 3)
Quiz 2	31/08/2024 7:00 PM	01/09/2024 7:00 PM	5 (Best 2 of 3)
Quiz 3	07/09/2024 7:00 PM	08/09/2024 7:00 PM	5 (Best 2 of 3)
Assignment 1	20/06/2024	18/07/2024	15
Assignment 2	25/08/2024	22/09/2024	15

Session Content but before that

Webinar will tentatively be on these dates:

19-Jun-24

10-Jul-24

14-Aug-24

4-Sep-24

Session Content

- Importance of text pre-processing
- Overview of past learnings (NLP)
 - Stemming
 - Tokenization
 - POS Tagging
- Text to Speech
 - Understanding T2S algorithms basic layer
 - Different algorithms working
 - Code and example via different libraries
 - Importance of T2S Synthesis, ASR etc.

Importance of text pre-processing

- **Text pre-processing** is a crucial step in Natural Language Processing (NLP) that involves cleaning and preparing text data for analysis.
- It includes techniques such as:
 - Tokenization
 - Stop Word Removal
 - Stemming
 - Lemmatization
 - Text Cleaning

Importance: These steps help in converting raw text into a more manageable and analyzable format, facilitating better model performance and more accurate results.

Tokenization

- Tokenization is the process of splitting text into smaller units called tokens, which can be words, phrases, or symbols.
 - Example: Input: "Natural Language Processing is fascinating."
 - "Output: ["Natural", "Language", "Processing", "is", "fascinating"]"
- Tools
 - NLTK: A leading platform for building Python programs to work with human language data.
 - SpaCy: An open-source software library for advanced NLP in Python.
 - Tokenizer APIs: Available in various programming languages and platforms.
- Importance: Tokenization is the first step in the text pre-processing pipeline and is crucial for the performance of subsequent steps.

Tokenization

Inconsistent Data Representation: Without tokenization, text data remains as large, unmanageable strings, making it difficult to analyze and process.

Model Performance: Machine learning models require numerical or categorical input. Without tokenization, converting text into a suitable format is impossible, leading to poor model performance.

Difficulty in Feature Extraction: Tokenization allows for extracting features such as word frequencies, n-grams, and more. Skipping this step hinders effective feature extraction.

Ineffective Text Cleaning: Tokenization is often the first step in text cleaning. Without it, removing stop words, punctuation, and performing stemming/lemmatization becomes challenging.

Error Propagation: Errors in initial steps propagate through the pipeline, leading to inaccuracies in tasks like sentiment analysis, NER, and POS tagging.

Stop-Words

- Stop words are common words that are usually ignored in text processing because they do not carry significant meaning.
 - Examples: Common stop words include "a", "the", "and", "in", "to", etc.
- Purpose:
 - Noise Reduction: Removing stop words helps in reducing the noise in the text data.
 - Efficiency: It reduces the size of the text data, making processing faster and more efficient.
- Tools:
 - NLTK: Provides a predefined list of stop words and functions for their removal.
 - SpaCy: Offers built-in support for stop word removal in various languages.
- Example:
 - Input: "The quick brown fox jumps over the lazy dog."
 - Output: "quick brown fox jumps lazy dog"
- Importance: Removing stop words helps in focusing on the words that are more likely to be significant in the analysis, thereby improving the performance of NLP models.

Stemming and Lemmatization

- Stemming: Reduces words to their base or root form by removing suffixes. It may not always produce a real word.
- Lemmatization: Converts words to their base form (lemma) using morphological analysis. It always returns a valid word.
- Examples:
 - Stemming: "running" -> "run", "jumps" -> "jump"
 - Lemmatization: "better" -> "good", "running" -> "run"
- Tools:
 - NLTK:
 - PorterStemmer for stemming
 - WordNetLemmatizer for lemmatization
 - **SpaCy**: Offers built-in lemmatization capabilities.

Stemming and Lemmatization

- Differences:
 - Accuracy: Lemmatization is generally more accurate than stemming.
 - Complexity: Stemming is simpler and faster but less accurate.
- Use Case: Choose stemming for quick and dirty text processing; use lemmatization for tasks requiring higher accuracy.
- Importance: Both techniques help in normalizing words to their base forms, which reduces the dimensionality of the text data and improves the performance of NLP models.
- Problems if we don't do stemming or lemmatization:
 - High dimensionality
 - Inconsistent Data
 - Reduced model performance and difficulty in text analysis
 - Lower accuracy in search and retrieval

Case Normalisation

- Case normalization is the process of converting all characters in the text to a uniform case, either lower case or upper case, to ensure consistency.
- Purpose:
 - Consistency: Ensures that words are treated equally regardless of their case.
 - Reduction of Redundancy: Helps in reducing redundancy by treating "Apple" and "apple" as the same word.
- Example:
 - Input: "Natural Language Processing"
 - Output: "natural language processing"
- Tools:
 - Python String Methods: `.lower()` and `.upper()`
 - NLTK: Provides functions for case normalization.
 - SpaCy: Built-in support for case normalization.
- Importance:
 - Improves Text Processing: Case normalization simplifies text processing by reducing the number of unique tokens.
 - Enhances Model Performance: Models become more efficient as they deal with fewer variations of the same word.
- Note: Case normalization is particularly useful when the case of the text does not carry significant meaning for the analysis.

Text Cleaning

- Text cleaning involves removing unwanted elements from the text to make it suitable for analysis.
- Common Techniques:
 - Removing Punctuation: Eliminating characters such as periods, commas, and exclamation marks.
 - Removing Special Characters: Removing symbols like #, \$, %, etc.
 - Removing Numbers: Excluding digits unless they are relevant to the analysis.
 - Removing HTML Tags: Stripping HTML content from web-scraped text.
 - Handling Contractions: Expanding contractions (e.g., "don't" to "do not").
- Tools:
 - Regular Expressions (Regex): Powerful for pattern matching and substitution.
 - NLTK: Provides functions for various text cleaning tasks.
 - SpaCy: Built-in functions for text cleaning.
- Example:
 - Input: "Hello, world! Visit us at <https://example.com> #NLP"
 - Output: "Hello world Visit us at example com NLP"
- Importance:
 - Enhances Data Quality: Cleaned text is more consistent and easier to analyze.
 - Improves Model Accuracy: Cleaner data leads to better-performing models by reducing noise and irrelevant information.

Named Entity Recognition (NER)



- Named Entity Recognition (NER) is a subtask of information extraction that identifies and classifies named entities in text into predefined categories such as person names, organizations, locations, dates, etc.

- **Categories:**

- **Person:** Names of people (e.g., "John Doe")
- **Organization:** Names of organizations (e.g., "Google")
- **Location:** Geographical locations (e.g., "Paris")
- **Date/Time:** Dates and times (e.g., "January 1, 2020")
- **Others:** Money, percentages, etc.

- **Purpose:**

- **Information Extraction:** Helps in extracting structured information from unstructured text.
- **Improved Search and Retrieval:** Enhances the performance of search engines and information retrieval systems.

- **Tools:**

- **SpaCy:** Provides a pre-trained NER model and tools for custom training.
- **NLTK:** Offers NER capabilities through its chunking module.
- **Stanford NER:** A widely used NER tool developed by Stanford University.

Importance: NER is essential for understanding the context and extracting relevant information from large volumes of text, making it a key component in various NLP applications.

Named Entity Recognition (NER)



Rule-based Approaches: Use predefined rules and patterns to identify entities.

- **Example:** Regular expressions to identify dates and email addresses.
- **Pros:** Simple to implement and interpret.
- **Cons:** Limited flexibility and scalability.

Machine Learning Approaches: Use statistical models trained on labeled data to identify entities.

- Conditional Random Fields (CRF)
- Hidden Markov Models (HMM)
- **Pros:** More flexible and accurate than rule-based approaches.
- **Cons:** Require labeled training data and computational resources.

Named Entity Recognition (NER)



- **Deep Learning Approaches:** Use neural networks to automatically learn features and patterns from data
 - Recurrent Neural Networks (RNN)
 - Long Short-Term Memory (LSTM)
 - Transformer-based models (e.g., BERT)
- **Pros:** High accuracy and ability to capture complex patterns.
- **Cons:** Require large datasets and significant computational power.
- **Example:**
 - **Input:** "Apple is looking at buying U.K. startup for \$1 billion."
 - **Output:** Entities: [("Apple", "ORG"), ("U.K.", "LOC"), ("\$1 billion", "MONEY")]

Importance:

Choosing the right technique depends on the specific requirements of the task, available data, and resources. Machine learning and deep learning approaches are preferred for their accuracy and scalability.

Part of Speech Tagging (POS)

- Part-of-Speech (POS) tagging is the process of assigning a part of speech to each word in a sentence. The parts of speech include nouns, verbs, adjectives, adverbs, pronouns, conjunctions, prepositions, and interjections
- **Understanding Syntax:** Helps in understanding the grammatical structure of sentences.
- **Disambiguation:** Resolves ambiguities by providing context to words (e.g., "book" as a noun vs. "book" as a verb).

Example The quick brown fox jumps over the lazy dog."

- POS Tags:
The (DT) quick (JJ) brown (JJ) fox (NN) jumps (VBZ) over (IN) the (DT) lazy (JJ) dog (NN)

Tools:

- **NLTK:** Provides a comprehensive POS tagging module.
- **SpaCy:** Offers efficient and accurate POS tagging capabilities.
- **Stanford POS Tagger:** A robust tool developed by Stanford University.

Importance:

POS tagging is fundamental for many NLP tasks such as parsing, text-to-speech conversion, and information extraction. It enables a deeper understanding of the syntactic and semantic properties of text.

Part of Speech Tagging (POS)

- **Rule-based Tagging:** Uses a set of hand-crafted rules to assign POS tags.
 - **Example:** "If a word ends in 'ly', tag it as an adverb (RB)."
- **Statistical Tagging:** Uses probabilistic models based on the likelihood of a word's POS tag given its context.
 - Hidden Markov Models (HMM)
 - Maximum Entropy Models
- **Machine Learning Approaches:** Use supervised learning techniques to predict POS tags.
 - Support Vector Machines (SVM)
 - Conditional Random Fields (CRF)
- **Deep Learning Approaches:** Uses neural networks to automatically learn features from data.
 - Recurrent Neural Networks (RNN)
 - Long Short-Term Memory (LSTM)
 - Transformer-based models (e.g., BERT)

Example:

- Sentence: "The cat sat on the mat."
- POS Tags: The (DT) cat (NN) sat (VBD) on (IN) the (DT) mat (NN)

Sentiment Analysis

- Sentiment analysis, also known as opinion mining, is the process of determining the sentiment expressed in a piece of text. It classifies text into positive, negative, or neutral sentiments.

Understanding Public Opinion: Helps businesses understand customer opinions and feedback.

Market Research: Analyzing trends and opinions about products, services, or events.

Social Media Monitoring: Tracking sentiment trends on social media platforms.

Applications:

Customer Feedback Analysis: Understanding customer satisfaction and improving services.

Brand Monitoring: Tracking public sentiment towards a brand.

Political Sentiment: Analyzing public opinion on political issues or candidates.

Product Reviews: Assessing the sentiment in product reviews to gauge consumer reactions.

Sentiment Analysis

Techniques:

Lexicon-based Methods: Use predefined lists of positive and negative words.

Machine Learning Approaches: Train classifiers (e.g., SVM, Naive Bayes) on labeled data.

Deep Learning Approaches: Use neural networks (e.g., LSTM, BERT) for higher accuracy.

Example:

Input: "I love the new design of your website!"

Output: Positive

Importance:

Sentiment analysis provides valuable insights into the emotions and opinions expressed in text, enabling better decision-making and strategy formulation.

Text to Speech Synthesis

- Text-to-Speech (TTS) synthesis is the process of converting written text into spoken words using computational methods.
- Purpose:
 - Accessibility: Provides access to information for visually impaired users.
 - User Experience: Enhances user interaction in virtual assistants and conversational agents.
 - Automation: Automates tasks that require reading text, such as news reading or announcements.
- Components:
 - Text Analysis:
 - Tokenization: Breaking text into smaller units such as sentences and words.
 - Linguistic Analysis: Determining the part of speech, phonetic transcription, and prosody (intonation, stress, rhythm).
 - Acoustic Modeling:
 - Phoneme Synthesis: Generating speech sounds based on phonetic transcription.
 - Prosody Generation: Adding natural intonation, stress, and rhythm to synthesized speech.
 - Speech Synthesis:
 - Concatenative Synthesis: Combining pre-recorded speech segments.
 - Formant Synthesis: Using mathematical models to generate speech sounds.
 - Waveform Synthesis: Using neural networks to generate high-quality, natural-sounding speech.

Text to Speech Synthesis

- Tools:
 - Google Text-to-Speech: A widely used TTS engine with high-quality voices.
 - Amazon Polly: A cloud-based service that converts text into lifelike speech.
 - IBM Watson Text to Speech: Provides a range of customizable voices.
 - Open Source Tools: eSpeak, Festival, and MaryTTS
 - Example:
 - Input: "Welcome to the world of Text-to-Speech synthesis."
 - Output: (Spoken audio)
 - **Importance:** TTS synthesis plays a crucial role in improving accessibility and enhancing user experiences across various applications and devices.
-

Text Generation Techniques

Text generation involves creating coherent and contextually relevant text from a given input or set of rules.

Key Techniques:

Rule-based Systems : Use predefined rules and templates to generate text.

Example: Fill-in-the-blank templates for automated report generation

Markov Chains : Use probabilistic models based on the likelihood of word sequences.

Example: Generating text by predicting the next word based on the previous one

Recurrent Neural Networks (RNNs): Use neural networks with loops to maintain context over sequences.

Example: Generating poetry or short stories

term

Long Short-Term Memory Networks (LSTMs) : A type of RNN designed to better handle long-dependencies.

Example: Generating more coherent paragraphs and articles.

Transformer Models: Use self-attention mechanisms to capture long-range dependencies in text.

Example: GPT-3 generating articles, stories, and dialogue.

State-based and Rule-based Dialogue Systems

Dialogue systems designed to follow specific states or rules to manage interactions with users.

State-based Dialogue Systems : Systems that manage conversations using predefined states and transitions between those states.

Rule-based Dialogue Systems: Systems that use predefined rules and templates to generate responses and manage interactions.

Examples : If user says "Hello", respond with "Hi, how can I help you today?"

If user asks for account balance, respond with "Please provide your account number."

Components:

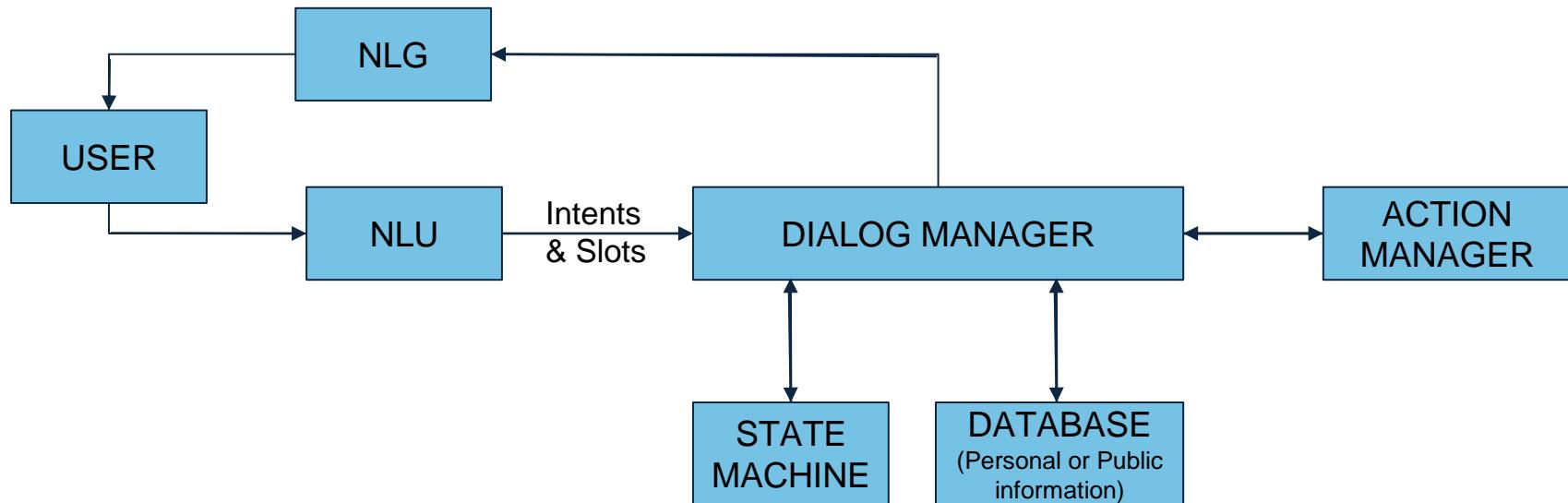
Rule Engine: Processes user input based on predefined rules.

Template Manager: Uses templates to generate responses.

Context Manager: Maintains context of the conversation to apply relevant rules.

State-based and Rule-based Dialogue Systems

State-based Dialogue Systems



Introduction to Intent Recognition and Slot Filling

Intent Recognition: The process of identifying the goal or purpose behind a user's input in a conversation.

Purpose: Helps in understanding user requests and guiding the conversation appropriately.

Example Intents:

Booking a flight

Slot Filling: Extracting specific pieces of information (slots) from the user's input that are necessary to complete the intent.

Purpose: Provides detailed information required to fulfill the user's request.

Example

Slots for Flight Booking:

Destination

Departure

DateReturn

Date

Number of Passengers

Introduction to Intent Recognition and Slot Filling

Components:

Natural Language Understanding (NLU) Module: Processes user input to recognize intents and extract slots.

Dialogue Manager: Uses recognized intents and filled slots to manage the conversation flow and fulfill the user's request.

Techniques:

Rule-based Methods: Use predefined patterns and templates to recognize intents and extract slots.

Machine Learning Approaches: Train classifiers on labeled datasets to predict intents and extract slots.

Deep Learning Approaches: Use neural networks, particularly sequence-to-sequence models, to handle more complex and varied inputs.

Example:

User Input: "I want to book a flight to New York on June 5th."

Recognized Intent: Book Flight

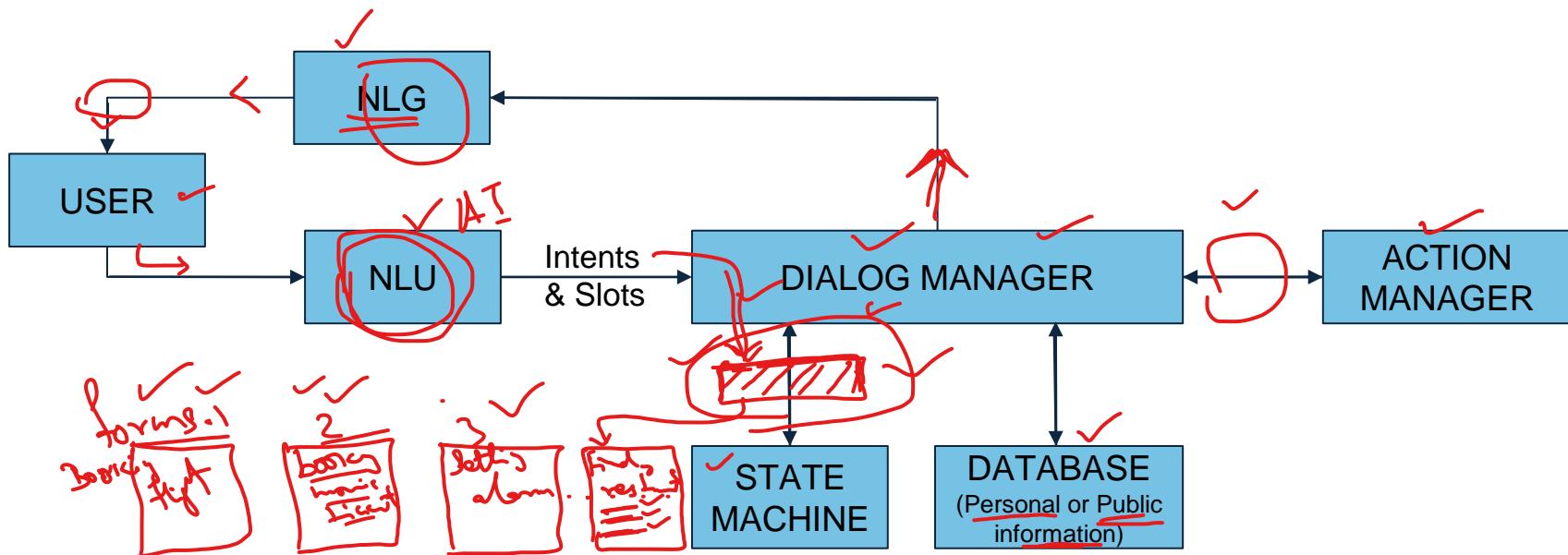
Extracted Slots:

Destination: New York

Departure Date: June 5th

State-based and Rule-based Dialogue Systems

State-based Dialogue Systems



Introduction to Intent Recognition and Slot Filling



ML ✓ [slot tagger]

Intent Recognition: The process of identifying the goal or purpose behind a user's input in a conversation.

Purpose: Helps in understanding user requests and guiding the conversation appropriately.

User: bot, please book ticket from delhi airport to bangalore airport

Example Intents:

Booking a flight

clarification from user:
→ Intent

Slot details:

Slot Filling: Extracting specific pieces of information (slots) from the user's input that are necessary to complete the intent.

Purpose: Provides detailed information required to fulfill the user's request.

Example

Slots for Flight Booking:

Destination

Departure

DateReturn

Date

Number of Passengers

ML

✓ book ticket for a man 28'.
✓ book ticket for a woman 28'.
✓ book ticket for a man 28'.
✓ book ticket for a woman 28'.

Introduction to Intent Recognition and Slot Filling

Components:

Natural Language Understanding (NLU) Module: Processes user input to recognize intents and extract slots.

Dialogue Manager: Uses recognized intents and filled slots to manage the conversation flow and fulfill the user's request.

Techniques:

Rule-based Methods: Use predefined patterns and templates to recognize intents and extract slots.

Machine Learning Approaches: Train classifiers on labeled datasets to predict intents and extract slots.

Deep Learning Approaches: Use neural networks, particularly sequence-to-sequence models, to handle more complex and varied inputs.

Example:

User Input: "I want to book a flight to New York on June 5th."

✓ Recognized Intent: Book Flight

Extracted Slots:

✓ Destination: New York

✓ Departure Date: June 5th

✓ from: ✓

dates [June] → Transformer ↗ ↘

11pm
cost

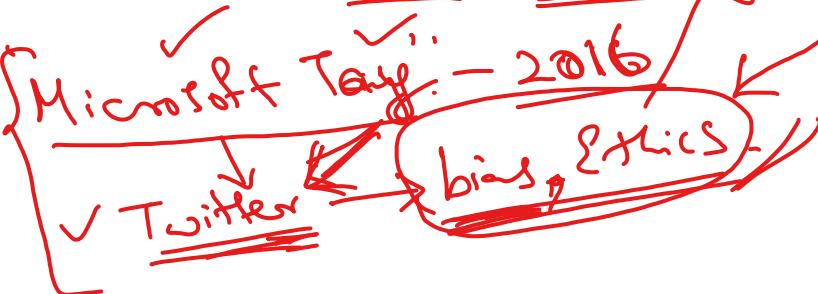
ML
6pm
cost

Key Components for Building Successful Bots (1 of 3)

Branding, Personality, and Human Involvement:

- ❖ Personality [Purpose, Target audience] → brand
- ❖ Logos and Icons [Alexa, Siri, ...] → brand
- ❖ Naming → brand, User understanding, memorable
- ❖ Human Intervention

When & how you need human to intervene



Artificial Intelligence (AI):

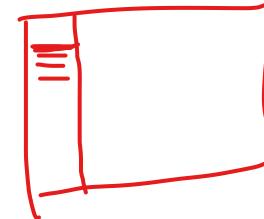
- ❖ Natural Language Understanding (NLU)
- ❖ Conversation Management
- ❖ Image Recognition
- ❖ Prediction
- ❖ Sentiment Analysis

delhi

Key Components for Building Successful Bots (2 of 3)

✓ Rich Interactions:

- ✓ ❖ Files ✓
- ✓ ❖ Audio and Video
- ✓ ❖ Images, Maps, and Charts
- ✓ ❖ Buttons ✓
- ✓ ❖ Templates ✓
- ✓ ❖ Links and Formatting ✓
- ✓ ❖ Emojis and Reactions
- ✓ ❖ Persistent Menus ✓
- ✓ ❖ Typing Indications ✓
- ✓ ❖ Slash Commands ✓
- ✓ ❖ Webviews ✓



- The Conversation:
- ❖ Onboarding
 - ❖ Functionality Scripting
 - ❖ Feedback and Error Handling
 - ❖ Help and Support
- (Handwritten annotations above the conversation list include: 'Greet & Purpose', 'Happy', and 'Error' with checkmarks.)*

Key Components for Building Successful Bots (3 of 3)

Context and Memory:

- ❖ Global Variables
 - ❖ Scoped Variables
 - ❖ Inferring Context from Pronouns
 - ❖ Deriving Context Through Rich Controls
 - ❖ Handling Context Errors
- User words* " ignore " *book my* *noise*
- [user queries]* *slot (state machine)*

Memory:

- ❖ Remembering Past Intents

- ❖ Association



Conversational AI

Introduction to LLMS

BITS Pilani
Pilani Campus

Kedar Kanhere
16/06/2024



Session Content

- Introduction to LLM's
- Fine tuning LLMs
- LLMs in Depth

Language Models

- The classic definition of a language model (LM) is a probability distribution over each token sequence

I w_1, w_2, \dots, w_n , whether it's a good or bad one.

- ~~Sally~~ fed my cat with meat: $P(I, \text{feed}, \text{my}, \text{cat}, \text{with}, \text{meat}) = 0.03$,
- My cat fed Sally with meat: $P(\text{My}, \text{cat}, \text{fed}, \text{Sally}, \text{with}, \text{meat}) = 0.005$,
- fed cat meat my my with: $P(\text{fed}, \text{cat}, \text{meat}, \text{my}, \text{my}, \text{with}) = 0.0001$

✓ $0.6 \leftarrow P(\cdot \cdot \cdot \text{The astronaut ate an } \underline{\text{apple}} \text{ in the } \underline{\text{Space}} \cdot \cdot \cdot)$
 $0.001 \leftarrow P(\cdot \cdot \cdot \text{...} \cdot \cdot \cdot \text{asteroid} \cdot \cdot \cdot \cdot \cdot \cdot)$

Autoregressive language models

- The chain rule of probability:
- $P(\text{Sally, fed, my, cat, with, meat}) = P(\text{Sally})$

Tent generation.

~~"In Space, the astrophysicist can"~~

$$P(X \mid Y) \Rightarrow$$

$$P(\text{apple} \mid Y) = 0.6$$

$$P(\text{astern} \mid Y) = 0.01$$

- * $P(\text{fed} \mid \text{Sally})$
- * $P(\text{my} \mid \text{Sally, fed})$
- * $P(\text{cat} \mid \text{Sally, fed, my})$
- * $P(\text{with} \mid \text{Sally, fed, my, cat})$
- * $P(\text{meat} \mid \text{Sally, fed, my, cat, with})$

Conditional probability

$$p(w_1, w_2, w_3, \dots, w_N) = \\ p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) \times \dots \times p(w_N|w_1, w_2, \dots, w_{N-1})$$

Generation

- If we already have a good language model, a given text prompt $w_{[1:n]}$, and we want the model to generate a good sentence completion with the length of L: How to find $w_{[n+1:n+L]}$ with the highest probability?
- Enumerate over all possible combinations?

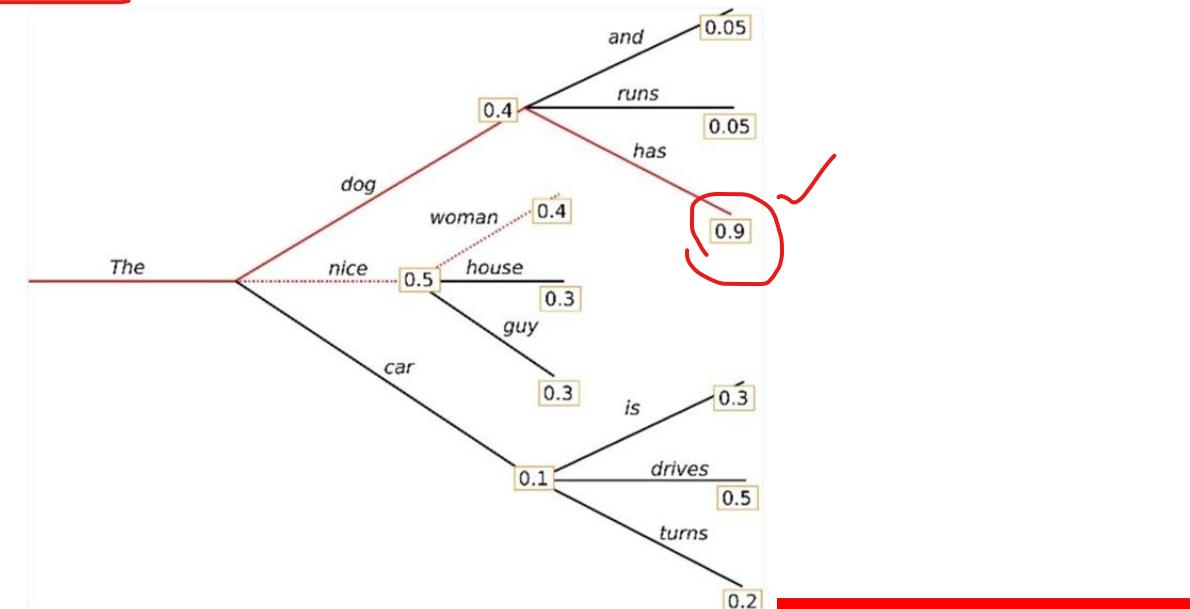
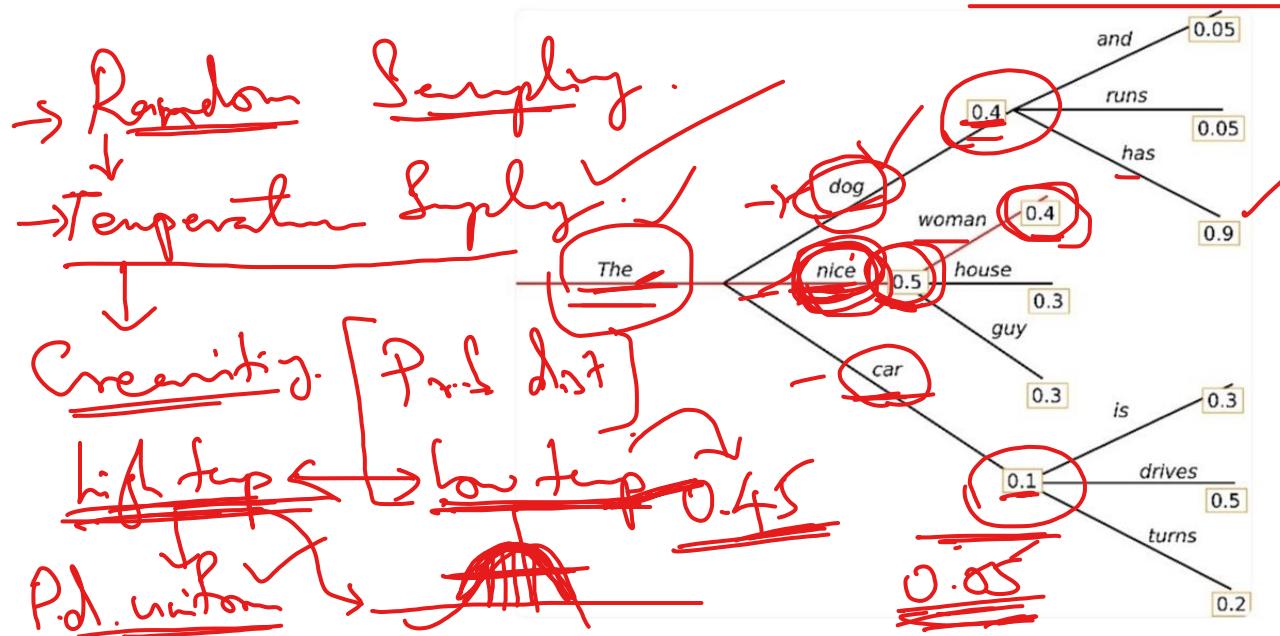
[whole class didn't read books] → Only she reads books.

[She doesn't read newspapers] → (She reads) only (books)

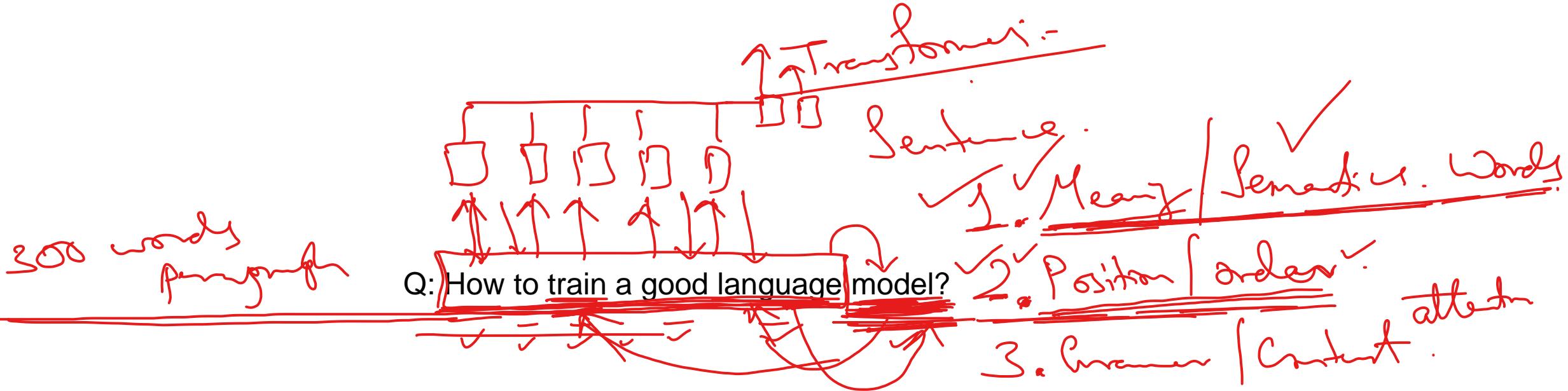
- Next token prediction: generating the next token step by step, starting from w_{n+1} using $p \frac{(w_{n+1} | w_{[1:n]})}{(w_{n+1} | w_{[1:n]})}$
- To select the next token with $p \frac{(w_{n+1} | w_{[1:n]})}{(w_{n+1} | w_{[1:n]})}$, there are also different decoding approaches.

Different Decoding Approaches

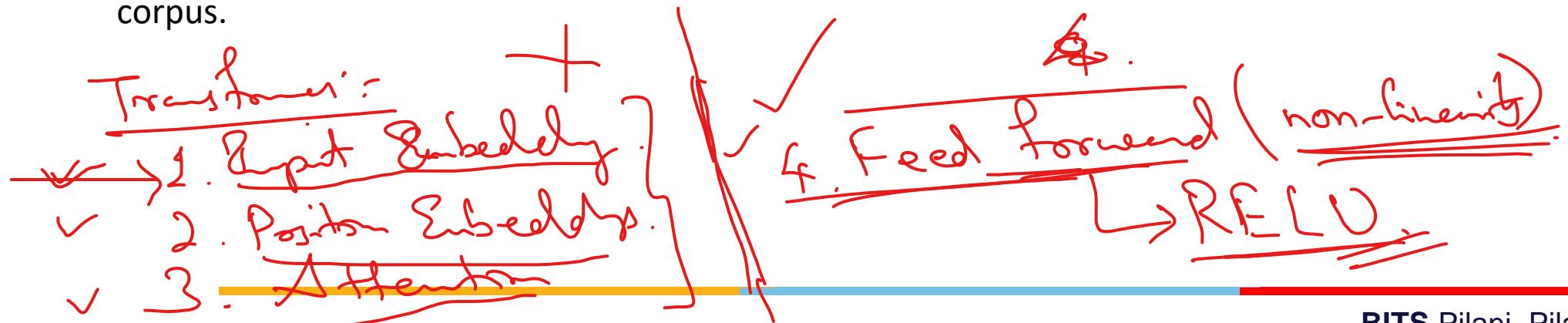
- lot of Computation, Overall missing local minima*
- Greedy decoding: At each step, always select w_t with the highest $p(w_t | w_{[1:t-1]})$
 - Beam Search: Keep track of k possible paths at each step instead of just one. Reasonable beam size k: 5-10.



Language models



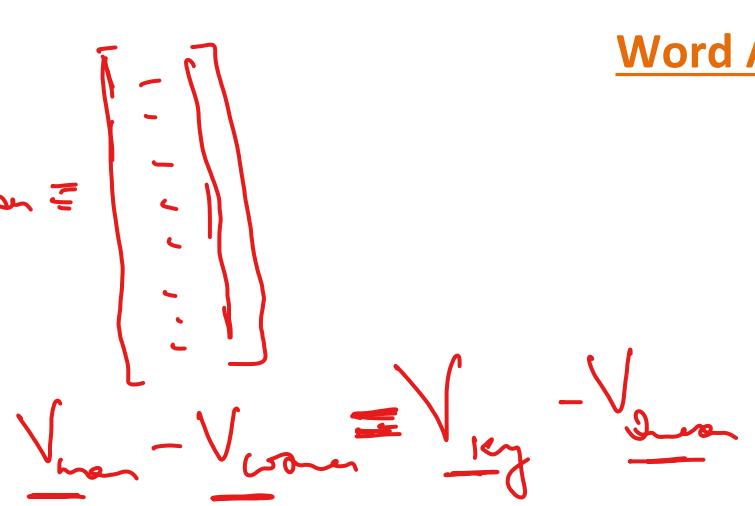
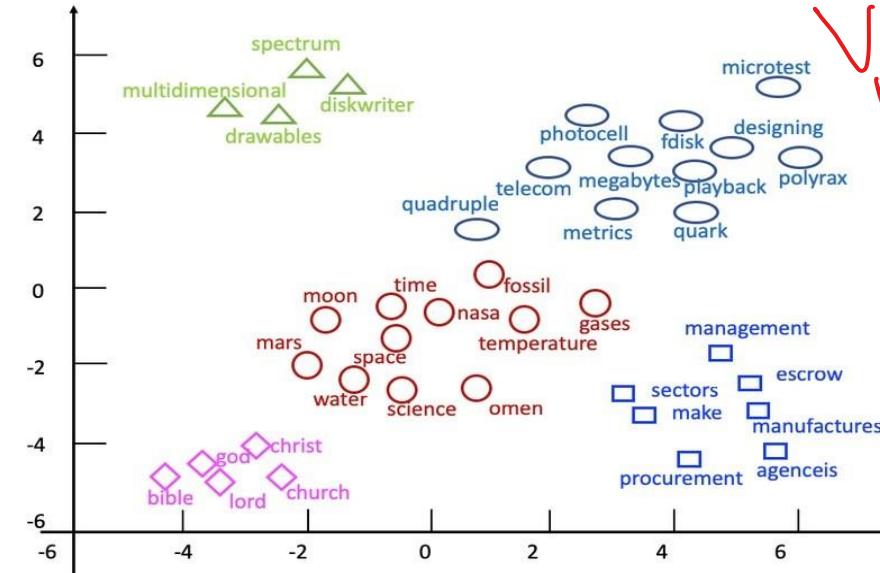
A: Maximizing the language model probability of an observed large corpus.



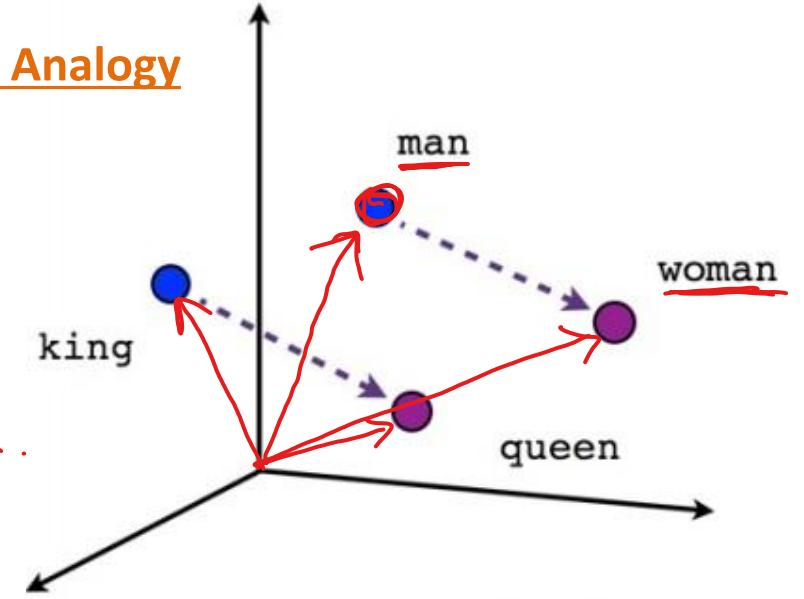
Representing Text in Language Models

- Unsupervised learning of text representations—No supervision needed
- Word embedding: Embed one-hot vectors into lower-dimensional space—Address “curse of dimensionality”
 - captures useful properties of word semantics
 - Word similarity: Words with similar meanings are embedded closer
 - Word analogy: Linear relationships between words (e.g. king - queen = man - woman)

Word Similarity



Word Analogy



Distributed Representations: Word2Vec

- Assumption: If two words have similar contexts, then they have similar semantic meanings!

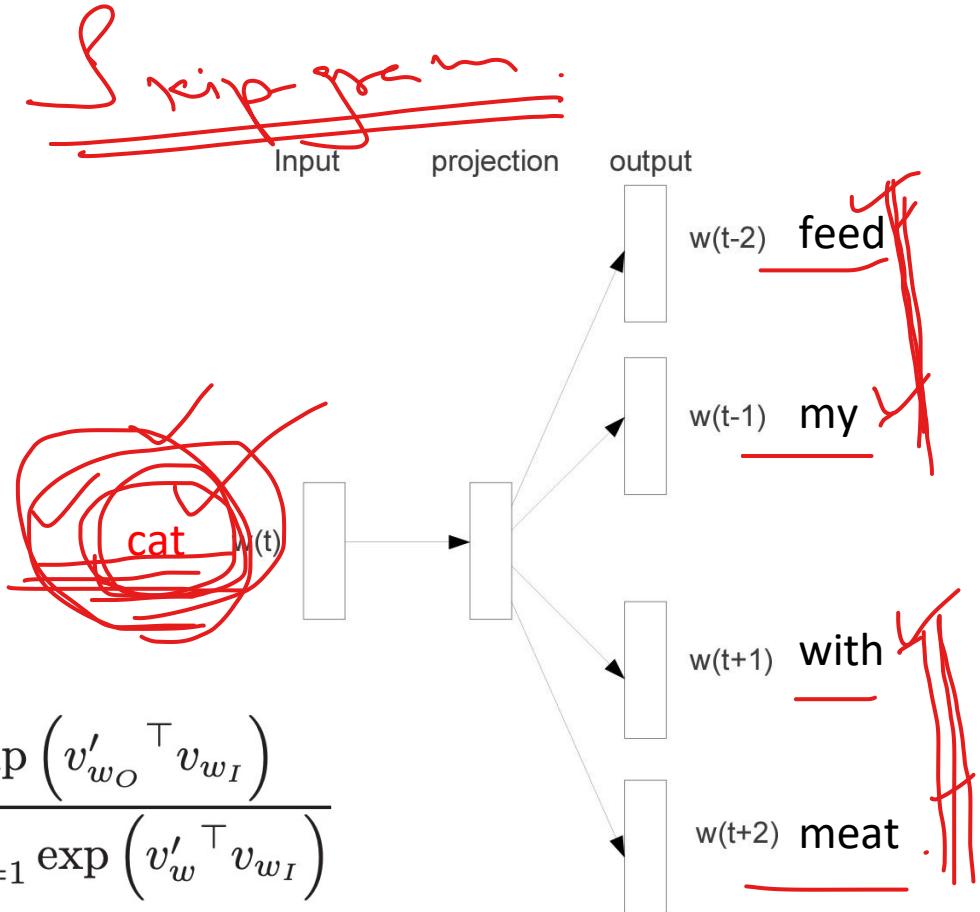
- Word2Vec Training objective:

- To learn word vector representations that are good at predicting the nearby words.

Co-occurred words in a local context window

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

$$p(w_O | w_I) = \frac{\exp(v'_{w_O}^\top v_{w_I})}{\sum_{w=1}^W \exp(v'_{w}^\top v_{w_I})}$$

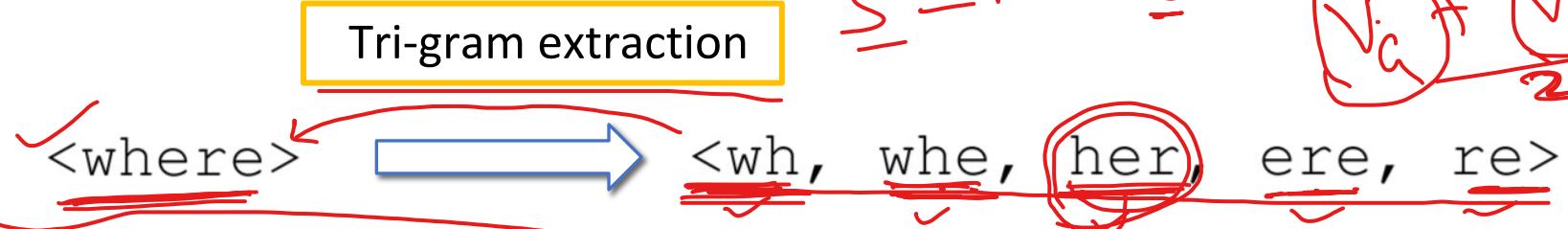


CBOW:

Considering subwords - fasttext

- fastText improves upon Word2Vec by incorporating subword information into word embedding

~~where
her~~



- fastText allows sharing subword representations across words, since words are represented by the aggregation of their n-grams

Word2Vec probability expression

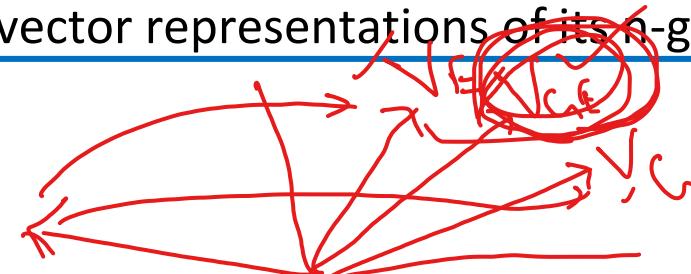
$$p(w_O | w_I) = \frac{\exp(v'_{w_O}^\top v_{w_I})}{\sum_{w=1}^W \exp(v'_{w'}^\top v_{w_I})}$$

~~New Vec, ect~~
~~Vector~~

N-gram embedding

Represent a word by the sum of the vector representations of its n-grams

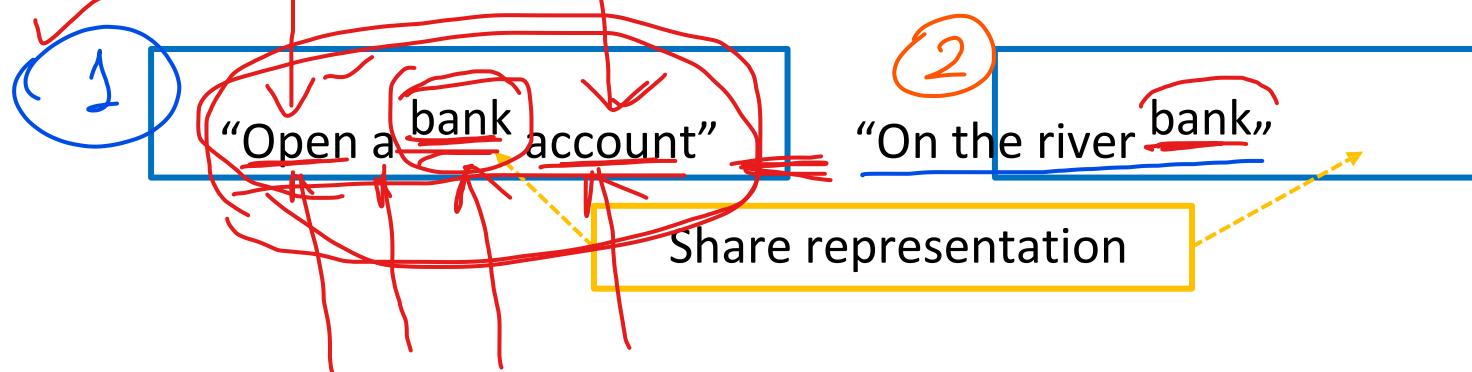
2. ~~running~~ ⇒ new word
→ $v_r + v_u \leftrightarrow v_{running}$



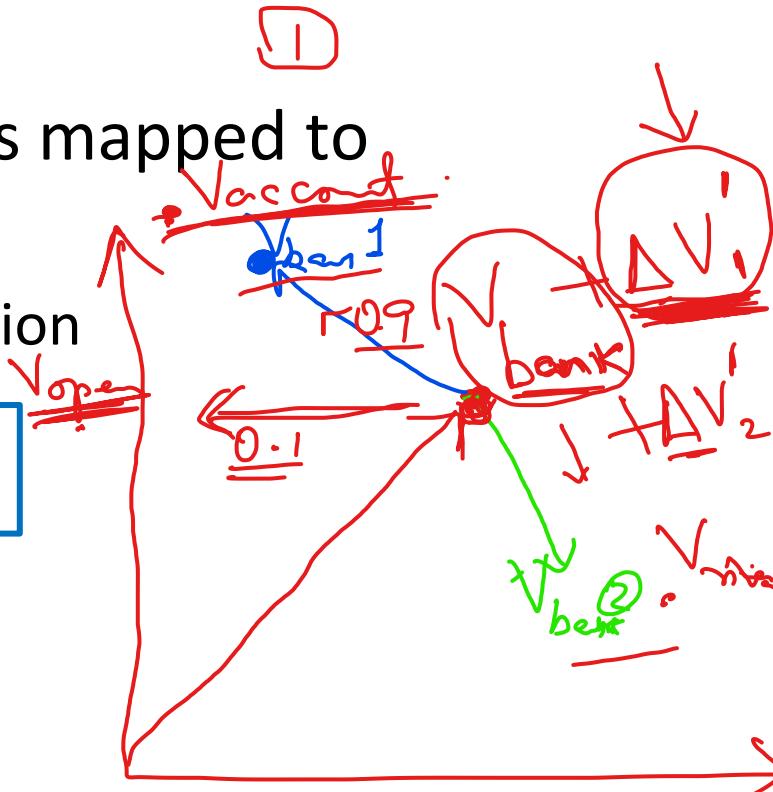
Limitations of embeddings

- 1) They are **context-free** embeddings: each word is mapped to only one vector regardless of its context!

- E.g. “bank” is a polysemy, but only has one representation

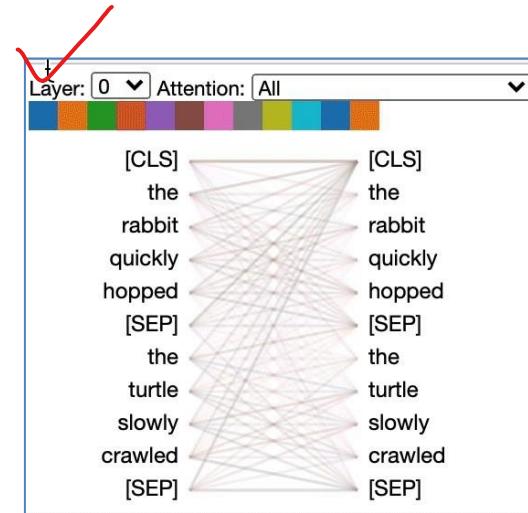
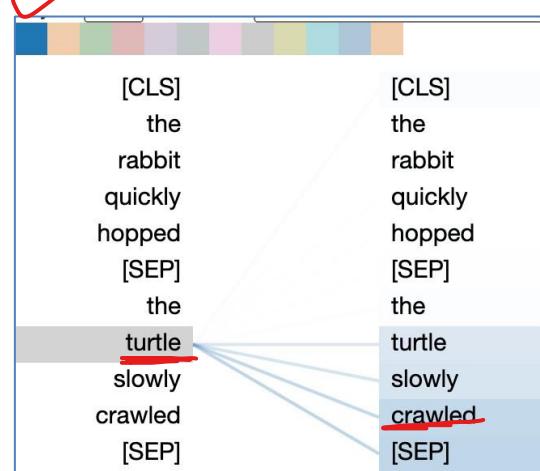
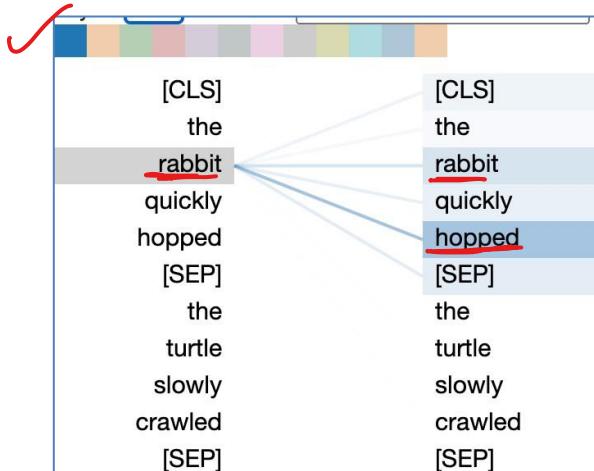


- ✓ 2) It does not consider the order of words
- ✓ 3) It treats the words in the context window equally



Attention is all you need

- Self-Attention: Each token attends to every other token in the sentence, but with different weights
- Demo: <https://github.com/jessevig/bertviz>



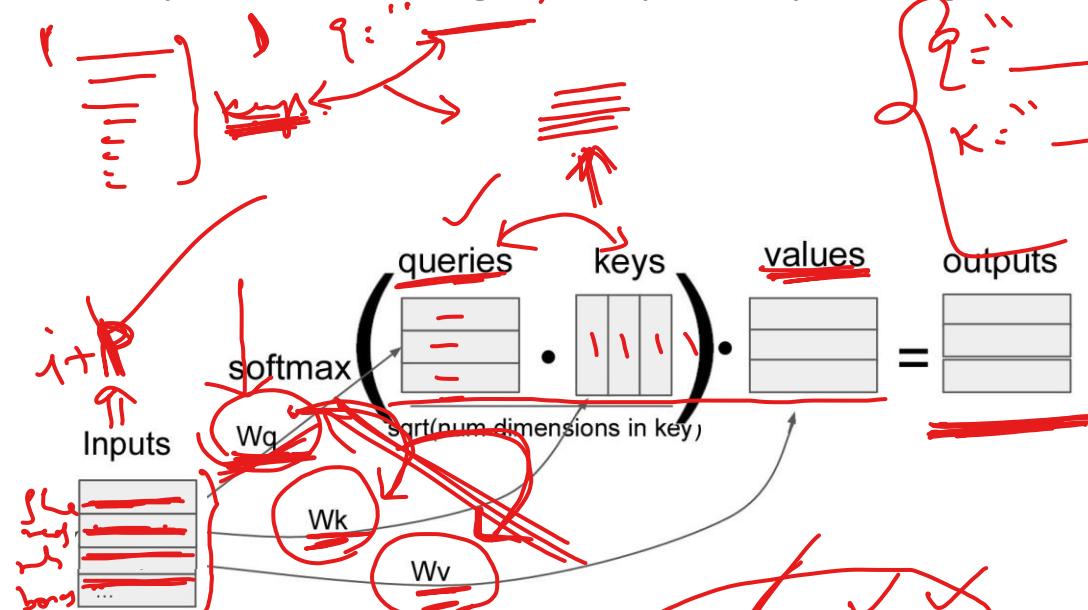
1. Embeddings (word)
2. Position Embeds.
3. Attention:
3.2 → Content (many).
3.5 → Weights to other words in the sentence.

Self-Attention

$$\Delta V_i + V_{\text{back}} \Rightarrow V_{\text{back}}$$

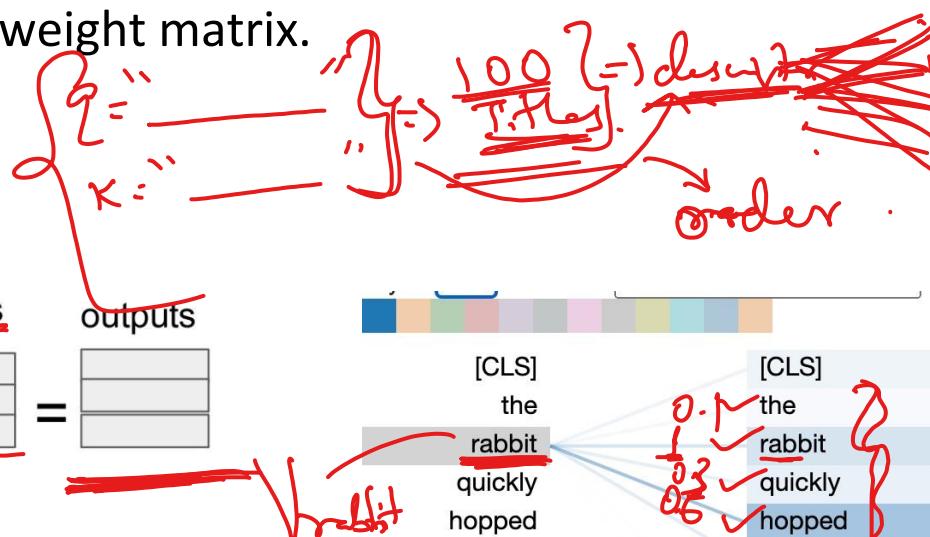
innovate achieve lead

- To calculate the attention weight from a query word w_q (e.g., "rabbit") to another word w_k
- Each word is represented as a query, key and value vector. The vectors are obtained from the input embeddings multiplied by a weight matrix.

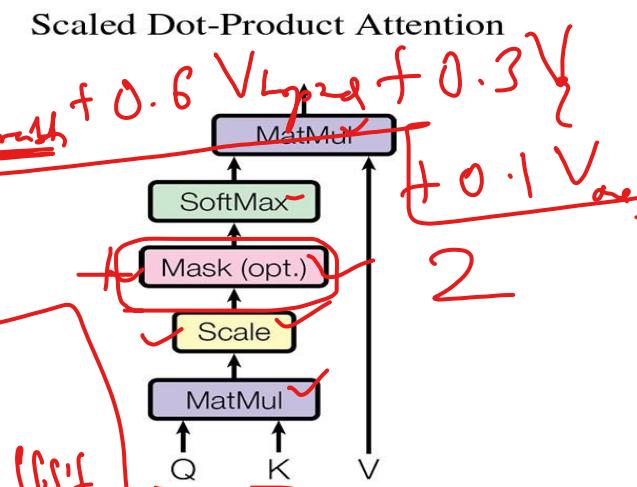
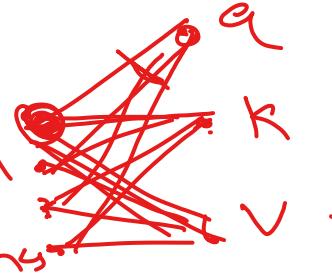


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$Q = (\text{inp}) \times \text{W}_q$$

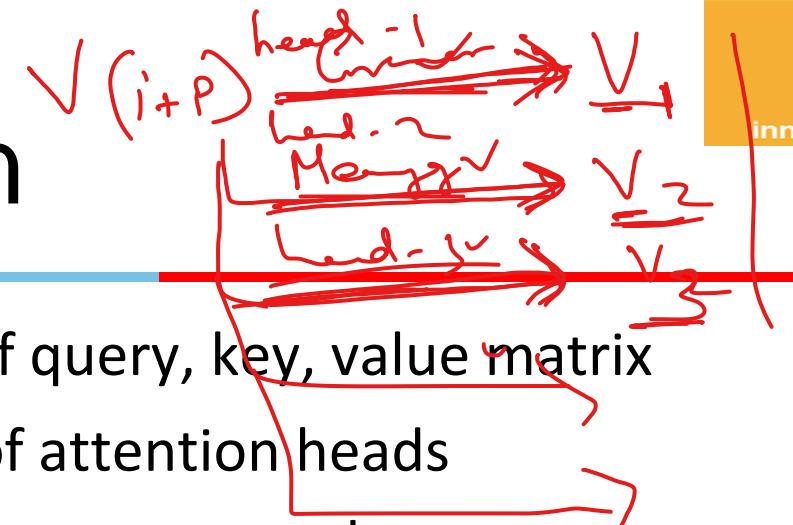


[CLS]	the	rabbit	quickly	hopped
	0.1	0.6	0.3	0.0
[SEP]	the	rabbit	quickly	hopped
	0.3	0.2	0.5	0.0
[SEP]	the	rabbit	quickly	hopped
	0.5	0.0	0.0	0.0



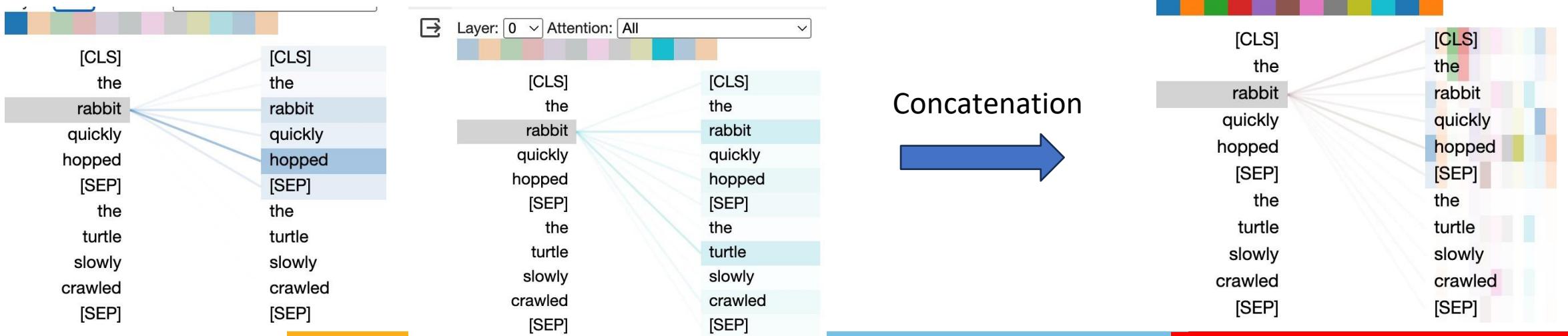
Multi-Head Attention

- Input: Multiple Independent sets of query, key, value matrix
- Output: Concatenate the outputs of attention heads
- Advantage: Each attention head focus on one subspace



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\underbrace{\text{head}_1, \dots, \text{head}_h}_{h=8}) W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$



LLM Pre Training Framework

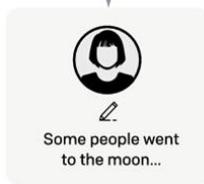
Step 1

Collect demonstration data, and train a supervised policy.

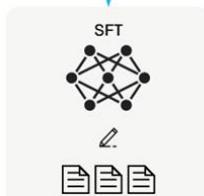
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

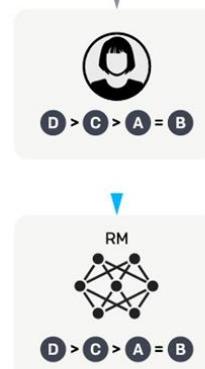


- (A) Explain gravity...
- (B) Explain war...
- (C) Moon is natural satellite of...
- (D) People went to the moon...

A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



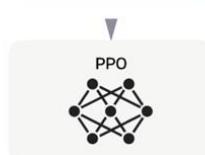
Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.



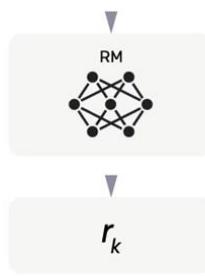
The policy generates an output.



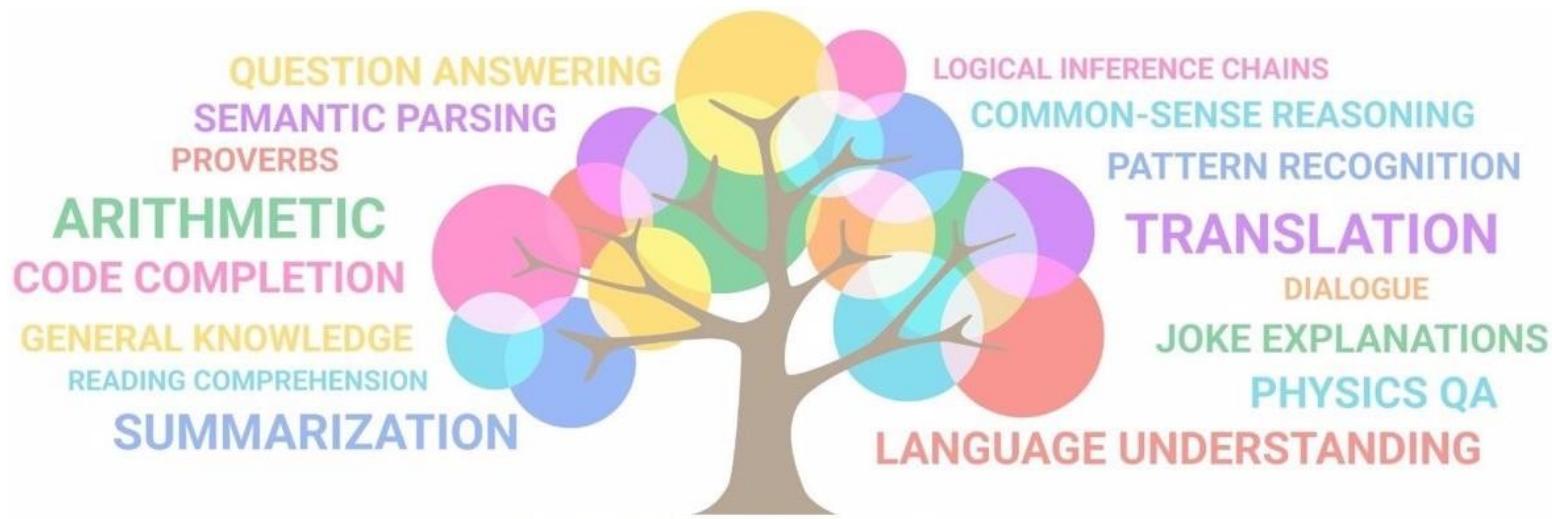
The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



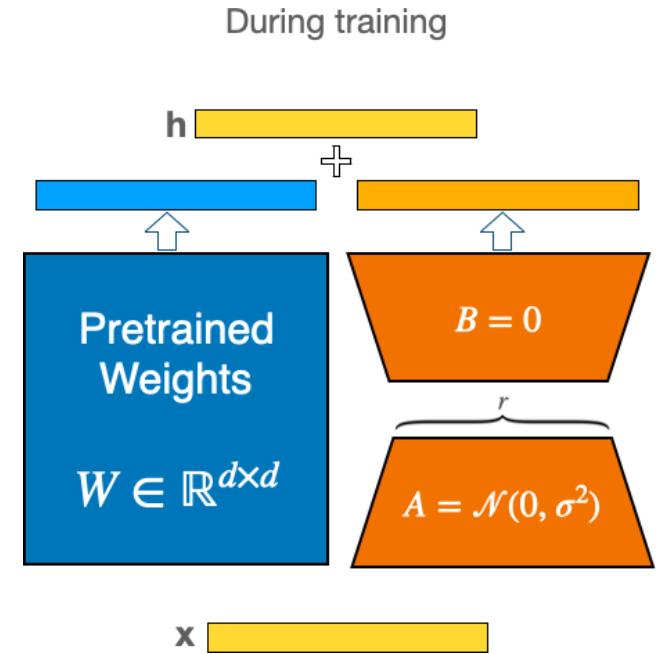
Efficient Fine-Tuning



Unsupervised/Self-supervised;
On large-scale general domain
corpus

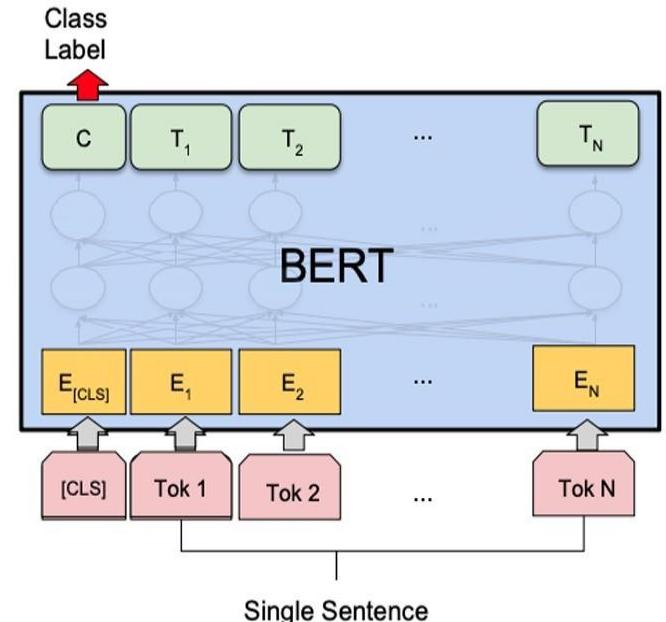


Task-specific
supervision; On target
corpus



Common NLP Tasks

- Single-sentence classification tasks
 - Text Classification Tasks
 - Input: The bike is too small and I want to return it.
 - Output: <refund, **return**, check_status>
 - Sentiment Analysis
 - Input: The restaurant is crowded and I waited my food for 30 minutes!
 - Output: <positive, **negative**>



Common NLP Tasks

- Sentence-pair classification tasks

- Sentence entailment

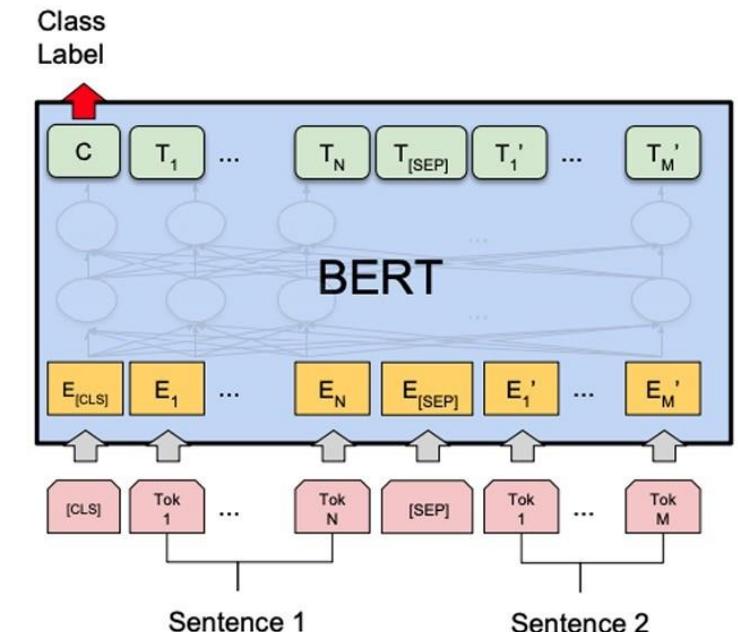
- Input:

Sentence 1: Our Large Language Model Course meets on Tuesdays and Thursdays in WashU.

Sentence 2: There is a large language model course in WashU.

- Output:

<entailment, contradiction, neutral>



Common NLP Tasks

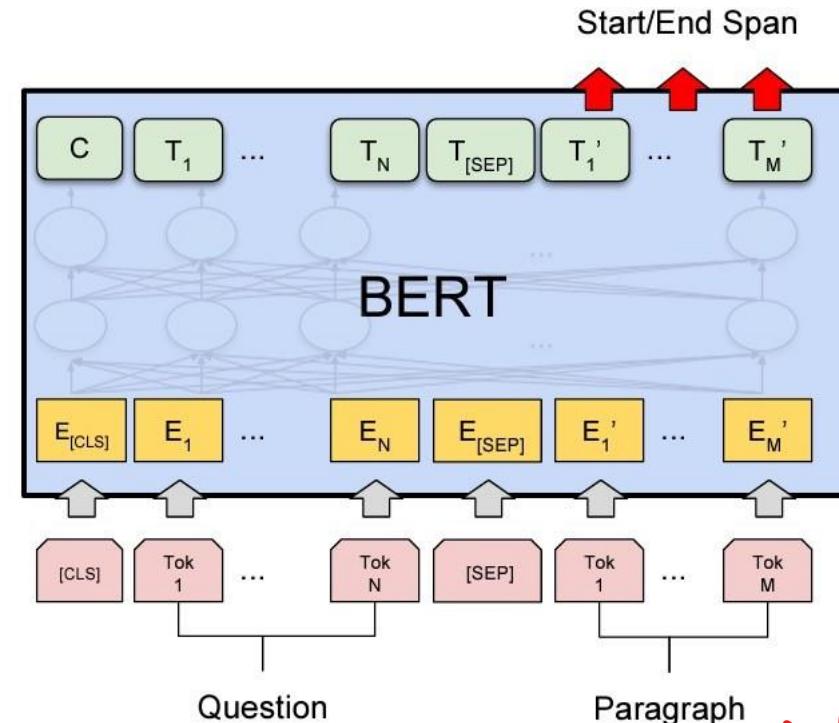
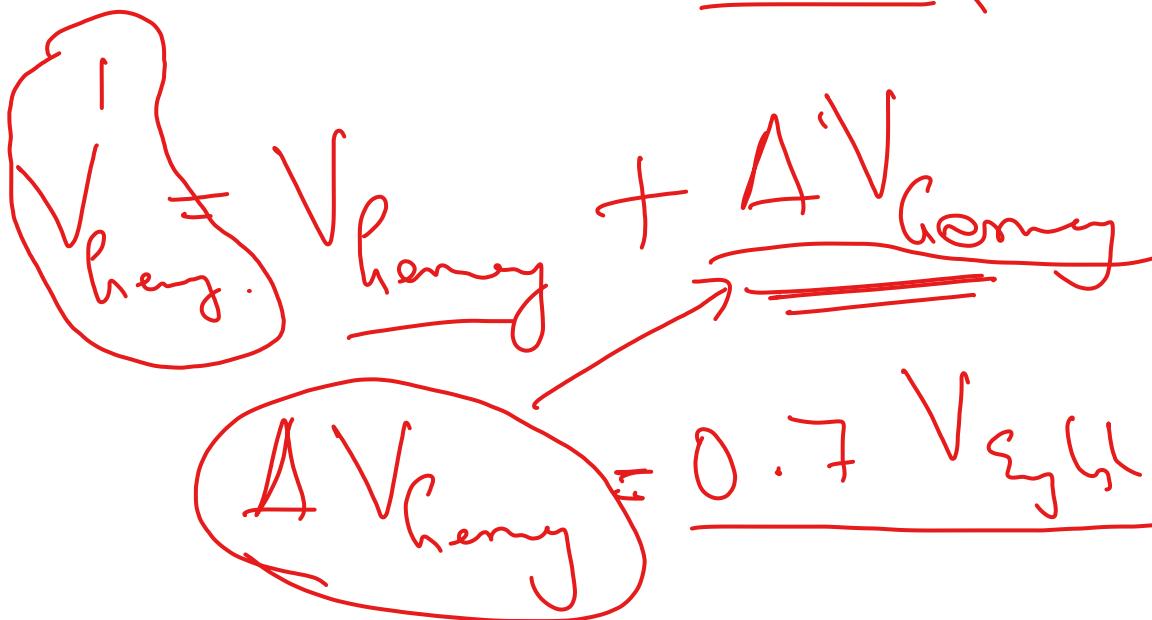
- Sentence generation tasks

- Machine Translation

- Input:

English: This is good. Germany:

- Output: Das ist gut.



Start/End Span

BERT

Question

Paragraph

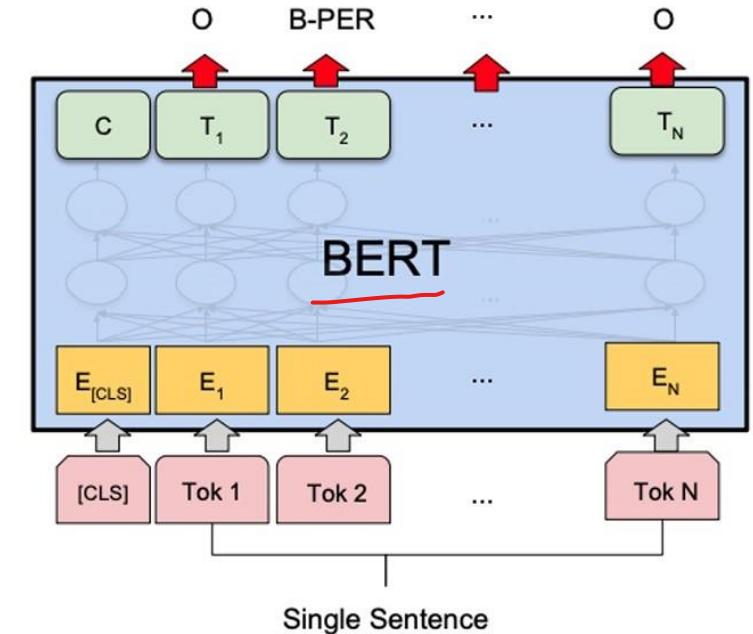
Start/End Span

Common NLP Tasks

New delhi
 ↓
 B-loc. Inside loc.

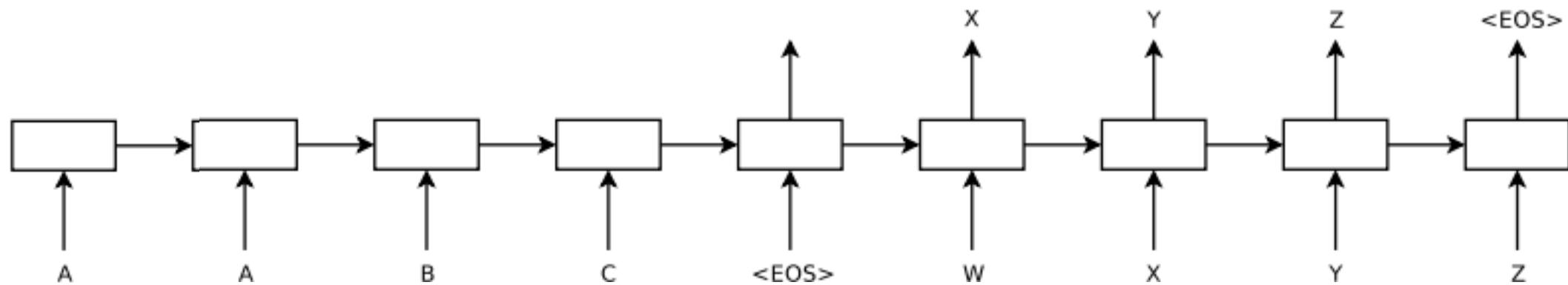
Birla 2st
 ↓
 I-Org. Inside org

- Token-level tasks
 - Named Entity Recognition
 - Input: St. Louis is located in the state of Missouri.
 - Output: <Begin-Location> <Inside-location> O O O O O O <Begin-Location> O



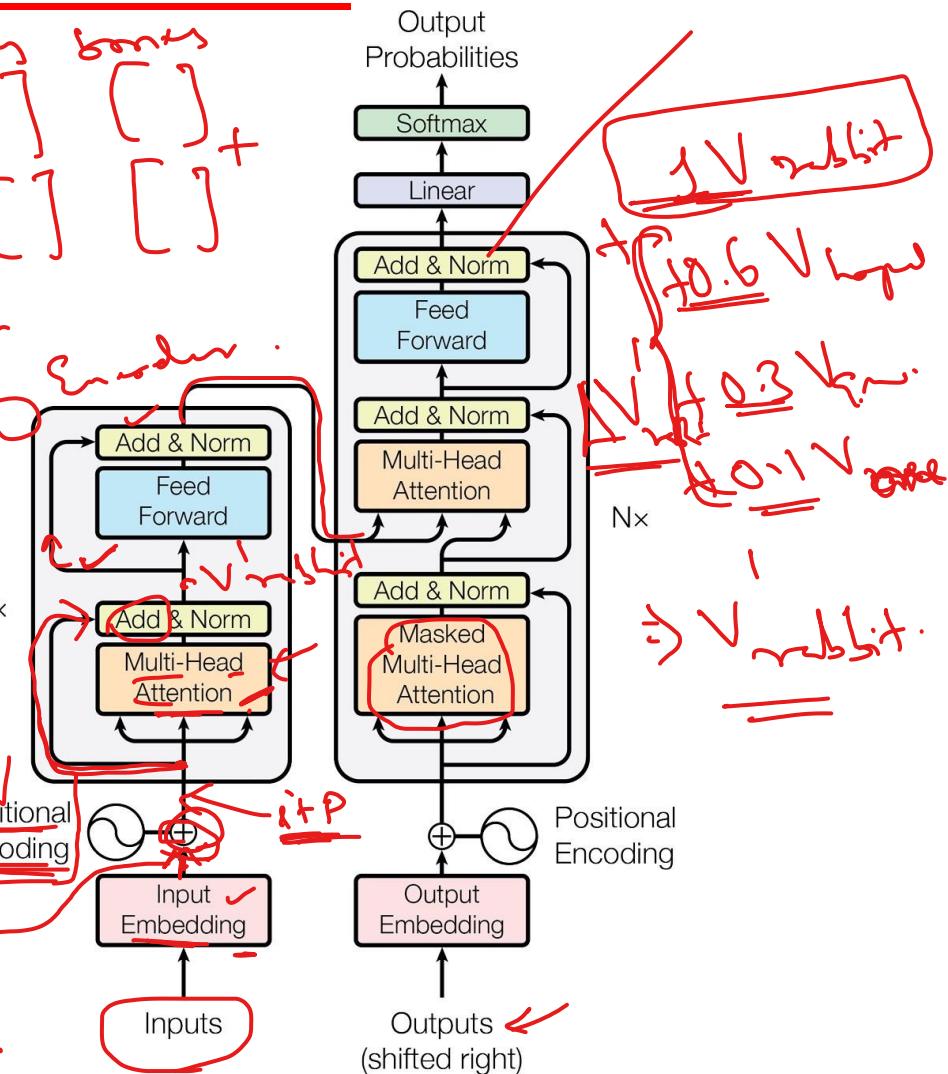
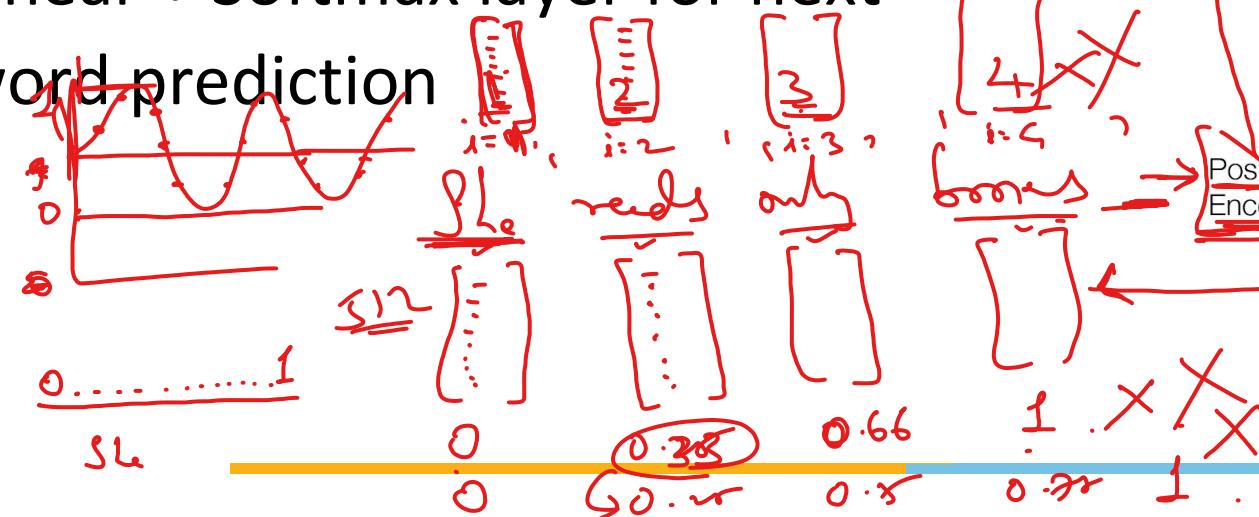
Encoder and Decoder

- NLP tasks can be generally decomposed into language understanding and language generation.
- Encoder models are generally used to understand input sentences, and decoder models are generally used to generate sentences.



Transformer Model Architecture

- Input Embedding
- Positional Encoding
- 12 Transformer layers
 - 6 encoder layers
 - 6 decoder layers
- Linear + Softmax layer for next word prediction



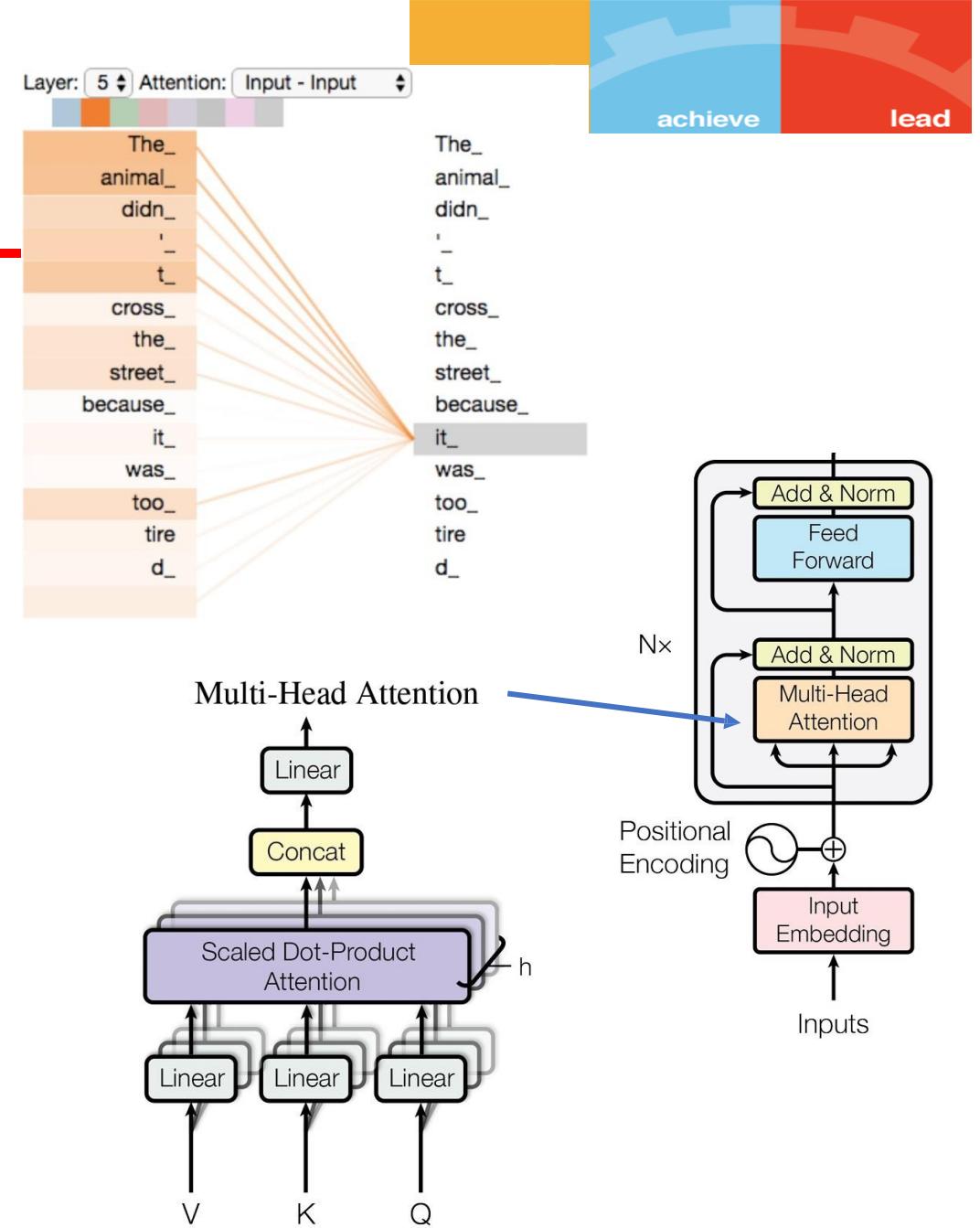
Encoder Model

- Multi-head attention layer captures information from different subspaces at different positions

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

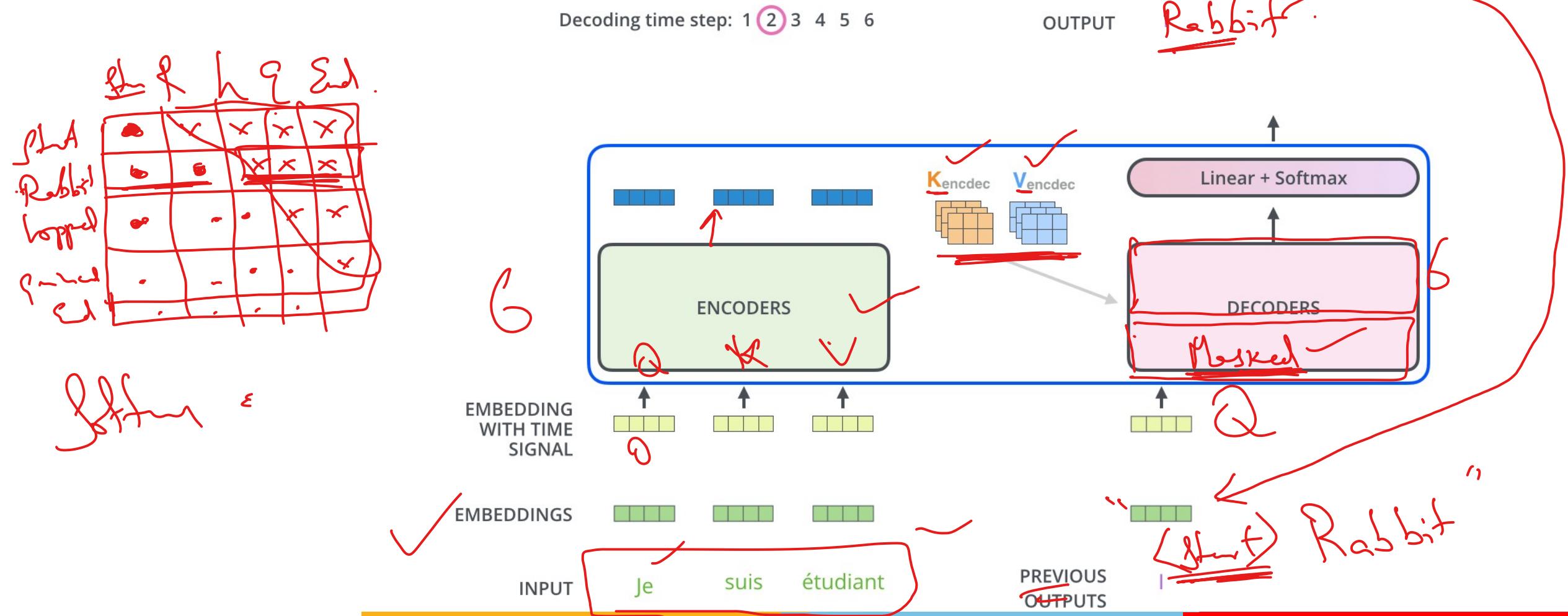
- Feed-forward layer is applied to each token position without interaction with other positions
- Residual connection and layer normalization





Decoder Model

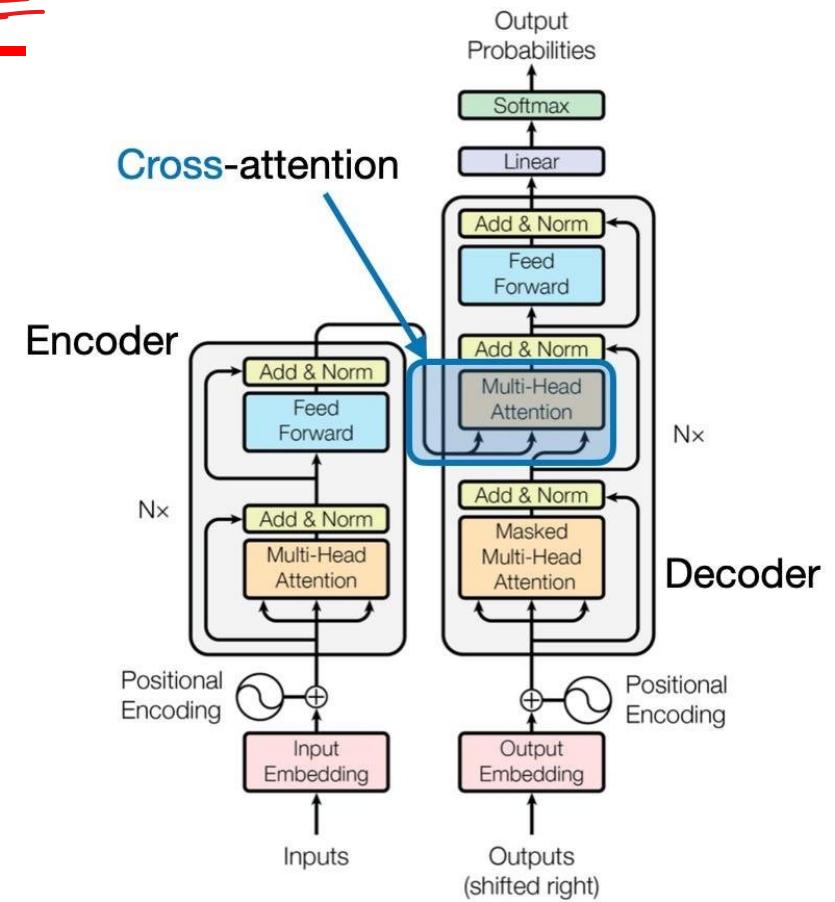
- Demo from <https://jalammar.github.io/illustrated-transformer/>



Decoder Model



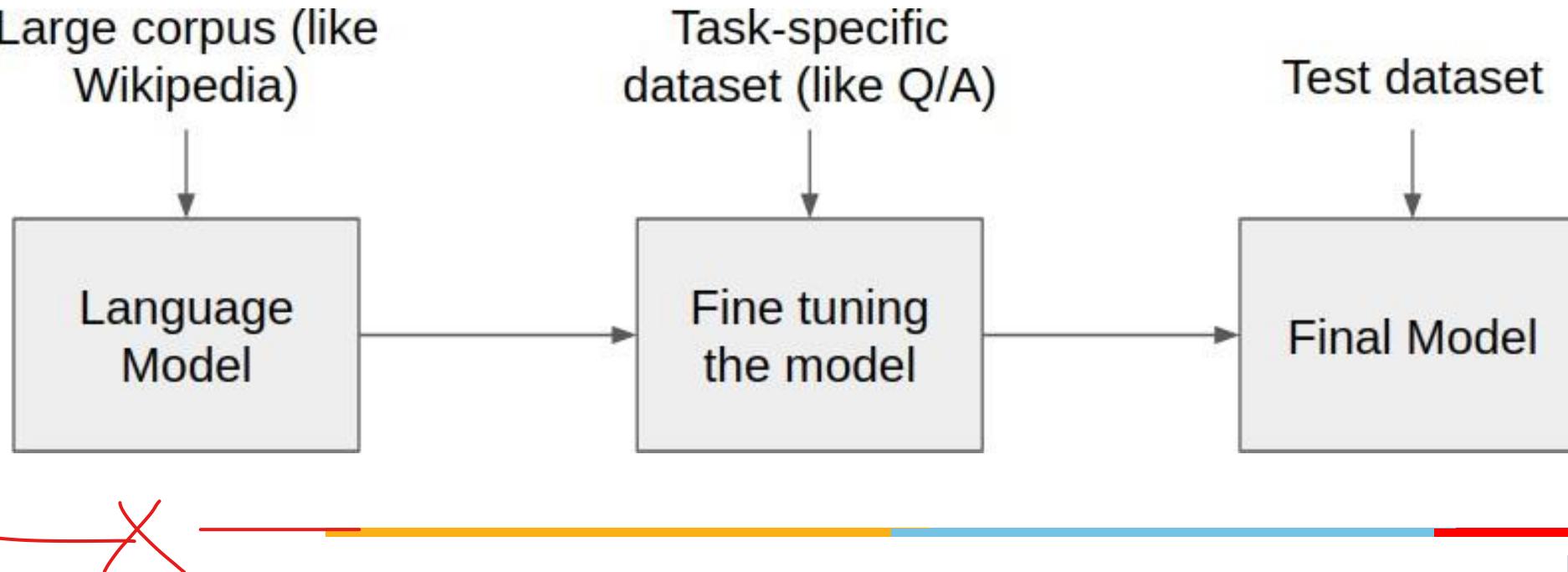
- Multi-head self-attention: only allowed to attend to earlier positions (left side).
 - Q, K, V matrices are both from the previously generated tokens
- Multi-head cross-attention: attend to the input sequence.
 - Q is from the generated tokens
 - K, V matrices are from the input tokens

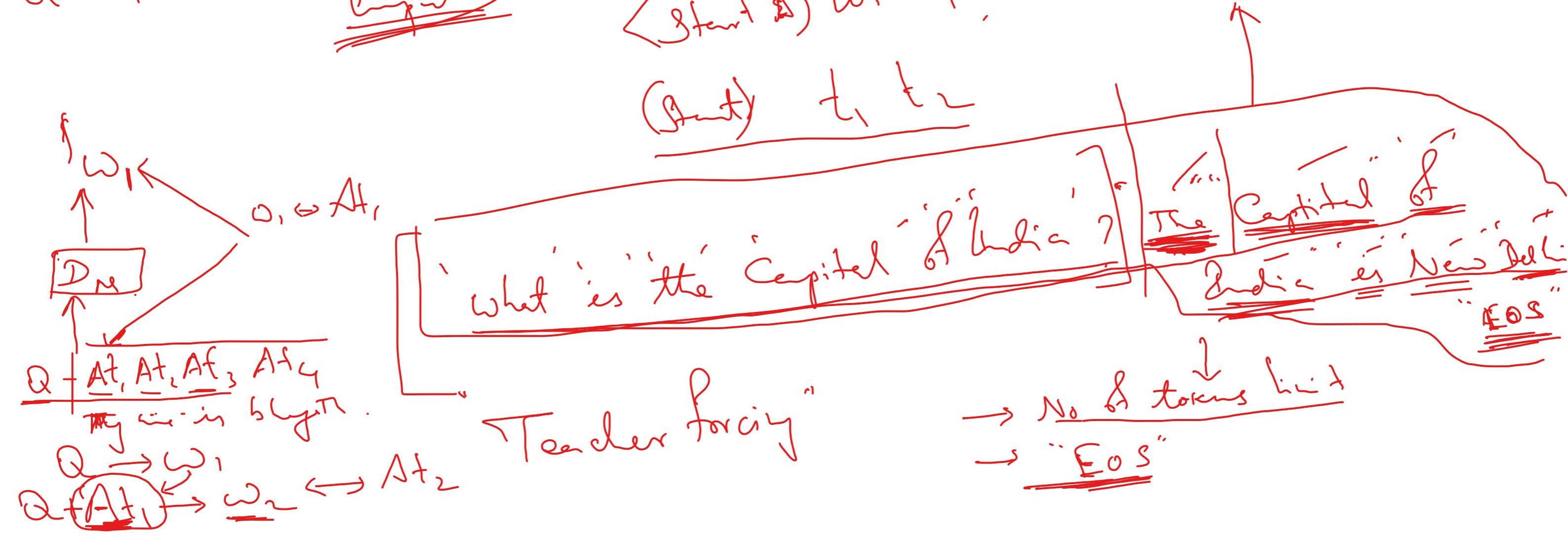
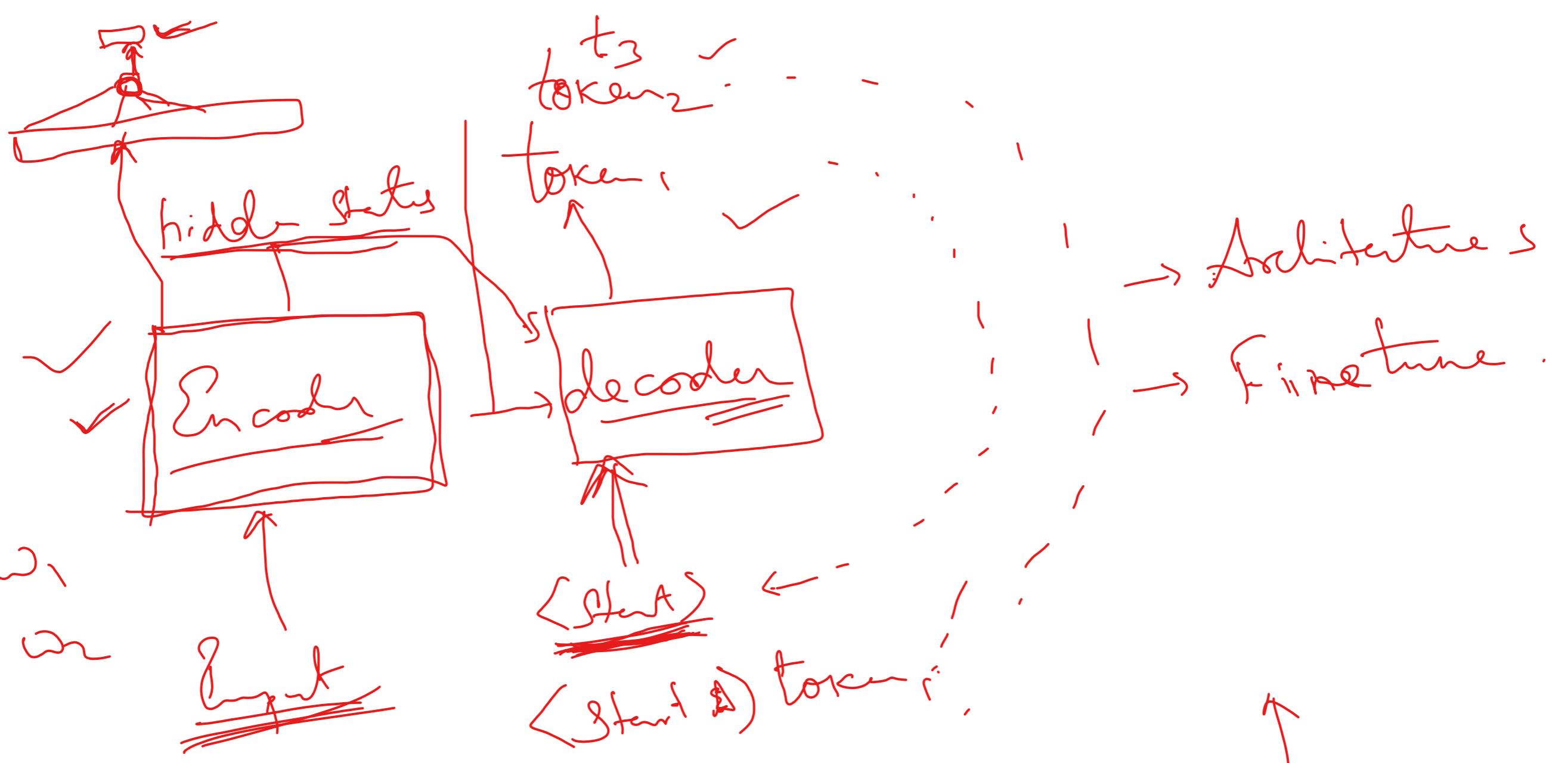


Source: "Attention Is All You Need" (<https://arxiv.org/abs/1706.03762>)

Pre-trained Language Models

- “Pretraining”: Train deep language models (usually Transformer models) via **self-supervised objectives** on **large-scale general-domain corpora**
- “Fine-tuning”: Adapt the pretrained language models (PLMs) to downstream tasks using task-specific data
- The power of PLMs: Encode generic linguistic features and knowledge learned through large-scale pretraining, which can be effectively transferred to the target applications





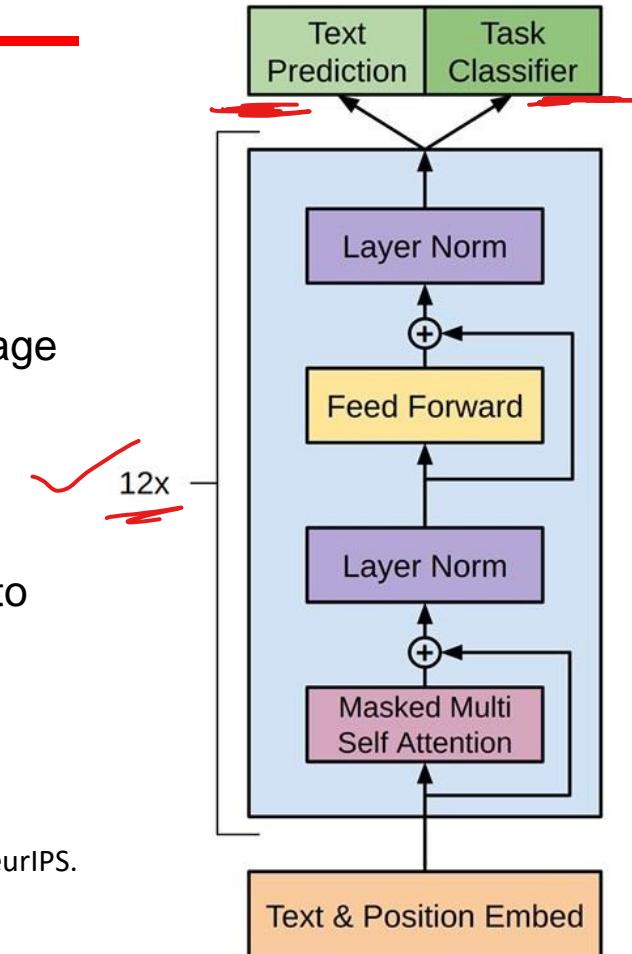
Different Architectures for PLMs

- ✓ • **Decoder-Only (Unidirectional) PLM** (e.g., GPT): Predict the next token based on previous tokens, usually used for **language generation tasks**
Causal
- ✓ • **Encoder-Only (Bidirectional) PLM** (e.g., BERT, XLNet, ELECTRA): Predict masked/corrupted tokens based on all other (uncorrupted) tokens, usually used for **language understanding/classification tasks**
- ✓ • **Encoder-Decoder (Sequence-to-Sequence) PLM** (e.g., T5, BART): Generate output sequences given masked/corrupted input sequences, can be used for both **language understanding and generation tasks**

Generative Pretraining (GPT)

- Model Architecture: A multi-layer transformer decoder
- Leverage unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

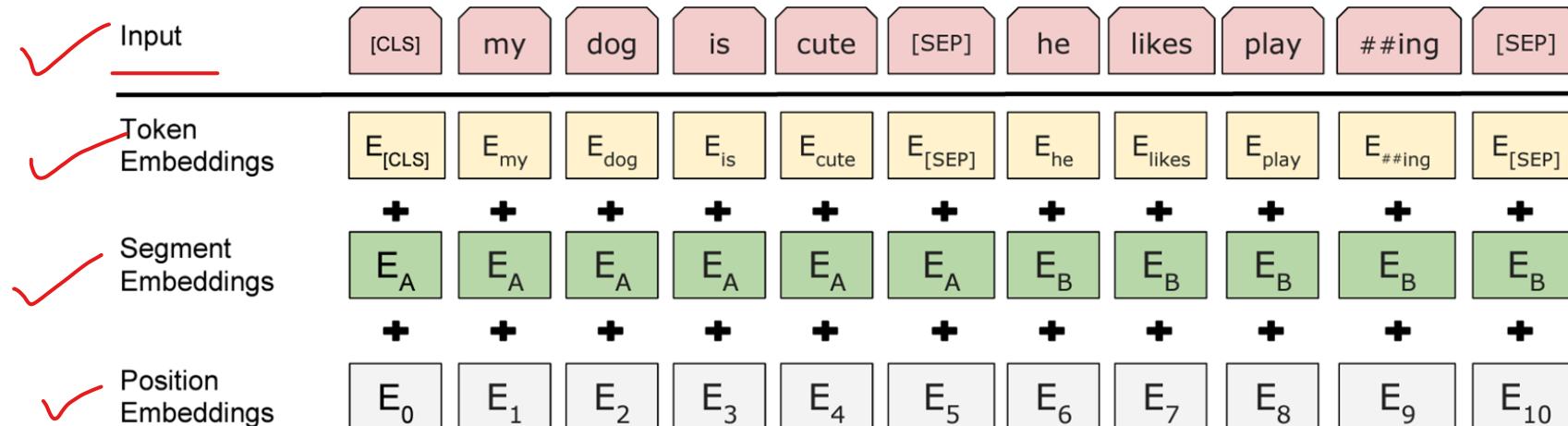
k previous tokens as context
- The Transformer uses **unidirectional** attention masks (i.e., every token can only attend to previous tokens)



1. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog.
2. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
3. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.

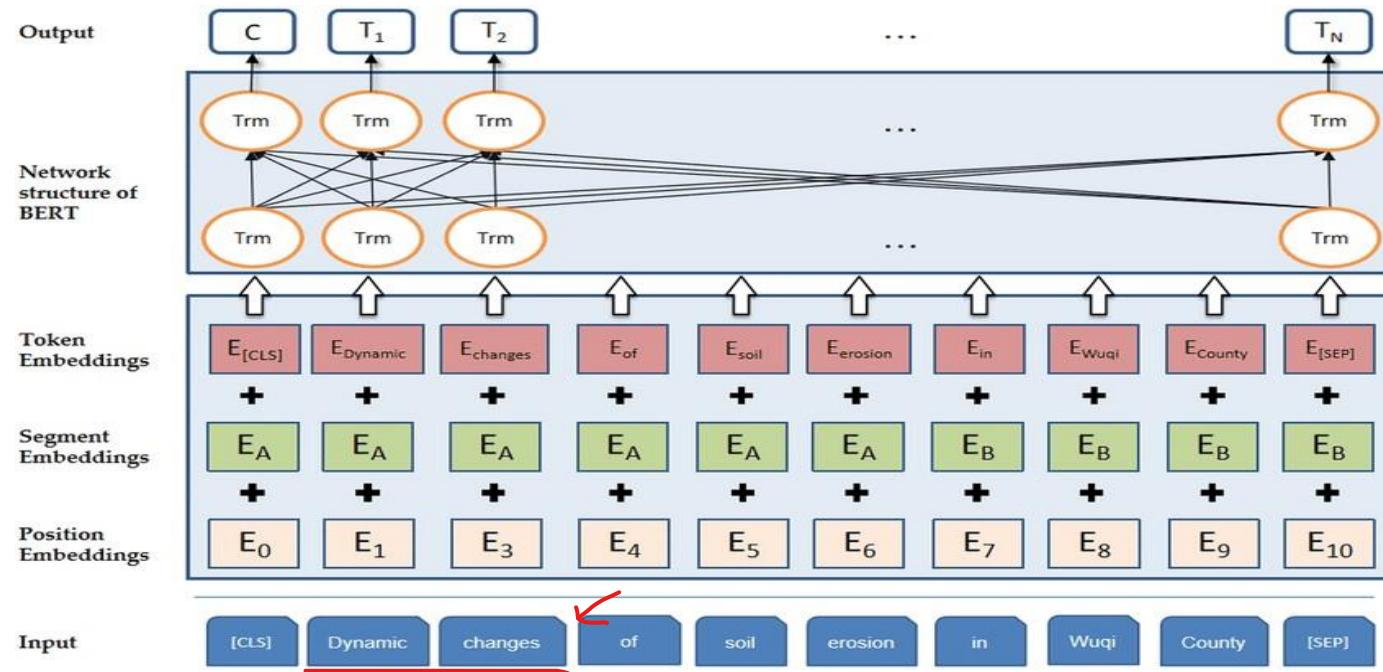
BERT Model Architecture

- Input: Sentence Pairs with special tokens [CLS] and [SEP].
 - Pair-wise tasks: question answering, translation, sentence entailment
 - [CLS]: beginning of a sentence
 - [SEP]: separation of two sentences
- WordPiece embedding



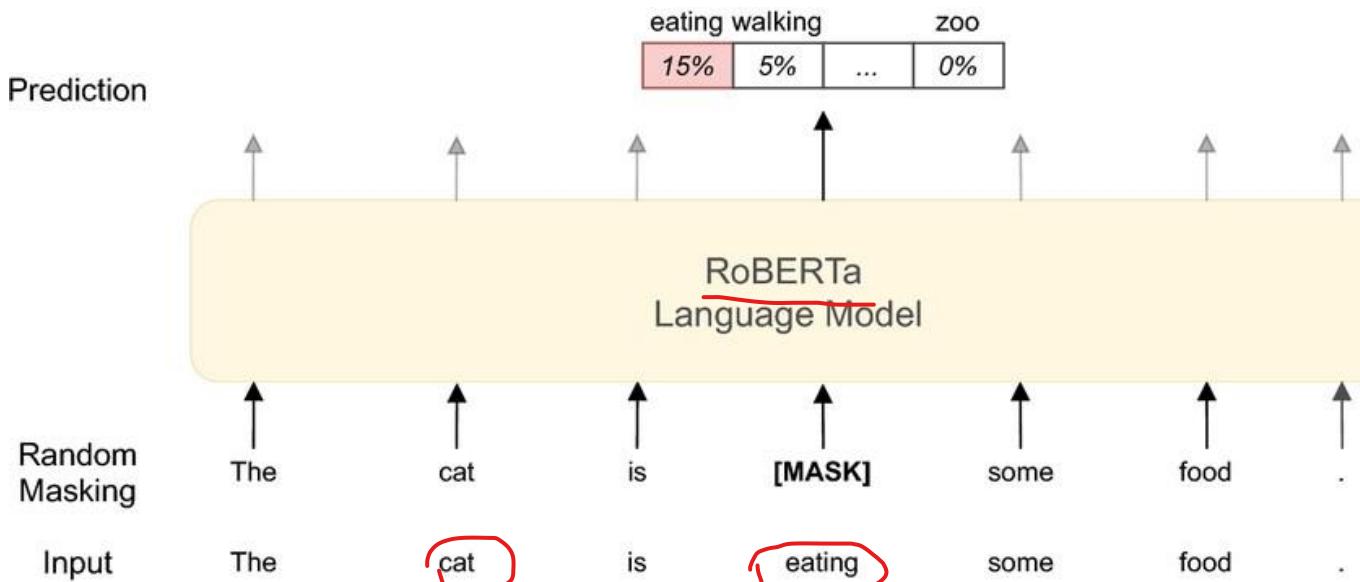
BERT Model Architecture

- Multi-layer Transformer encoder model
 - Bert-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters
 - Bert-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters
- Bidirectional: each token can attend to its left and right context for self-attention



BERT Training Objective (I): Masked Language Modelling

- Masked Language Modeling (MLM)
 - Randomly mask a few words in the original sentences.
 - Predict the masked words with its left and right context.
 - Mask ratio: 15%
 - Demo: <https://huggingface.co/bert-base-cased>

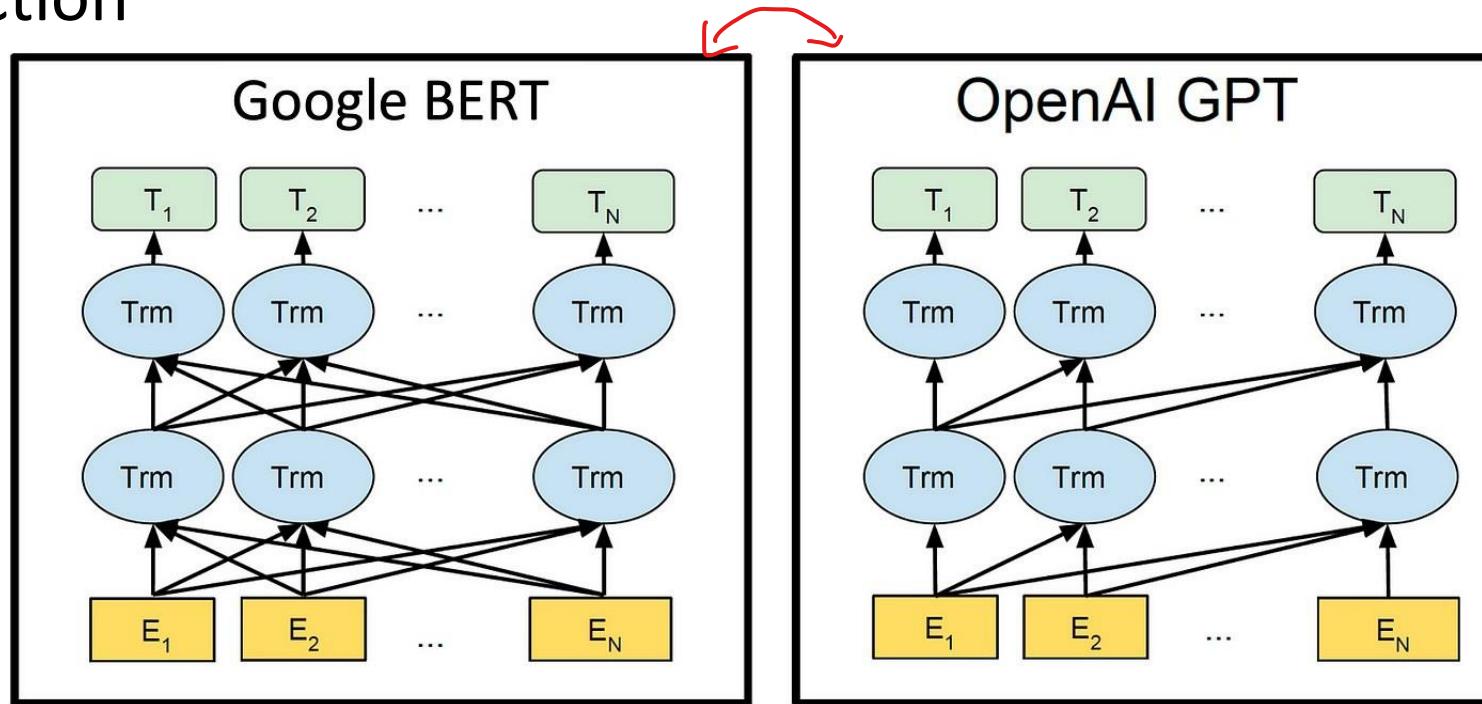


BERT Training Objective (I): Masked Language Modelling

- Discrepancy between pre-training and fine-tuning: There is no [MASK] word in a test set!
- 80-10-10 replacing rule for the sampled 15% tokens.
 - 80%: Replace the original word with [MASK] token
 - cat eating food -> cat [MASK] food
 - 10%: Replace the original word with a random token
 - cat eating food -> cat paper food
 - 10%: Do not replace the original word
 - cat eating food -> cat eating food

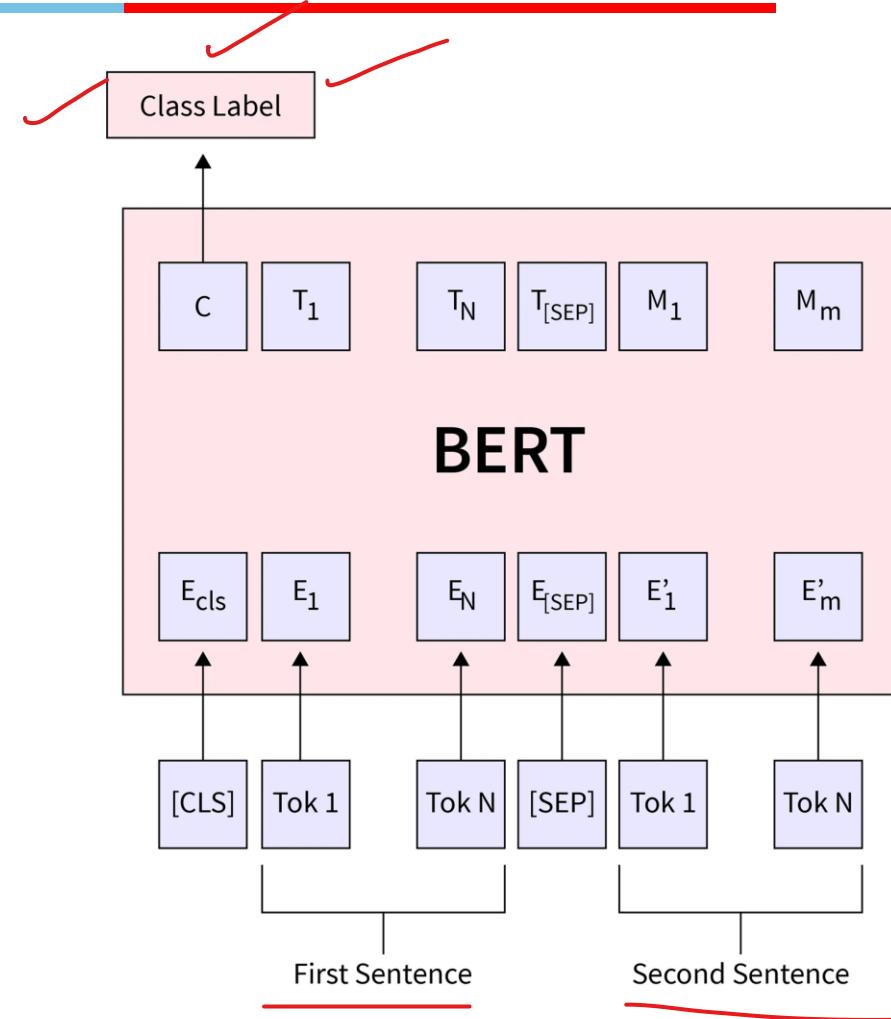
Comparison with GPT-2 Model

- Training objective: MLM prediction vs. left-to-right token prediction



BERT Training Objective (II): Next Sentence Prediction

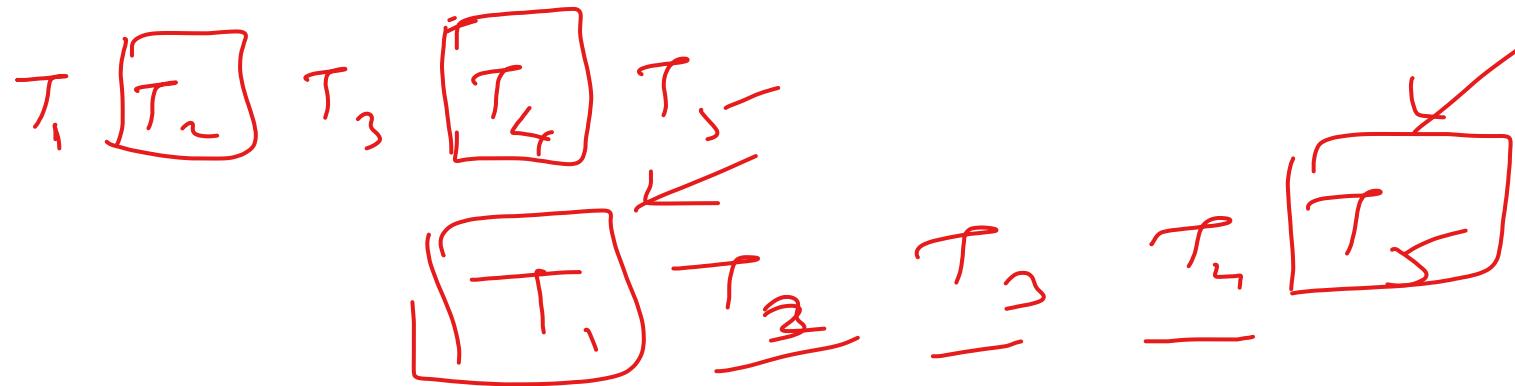
- Next Sentence Prediction (NSP)
 - Predict whether Sentence B is the next sentence of Sentence A.
 - Positive samples: two contiguous sentences in the corpus.
 - Negative samples: sample another sentence for sentence A.
 - Class Labels: <is_next, not_next>



Variants of BERT Model

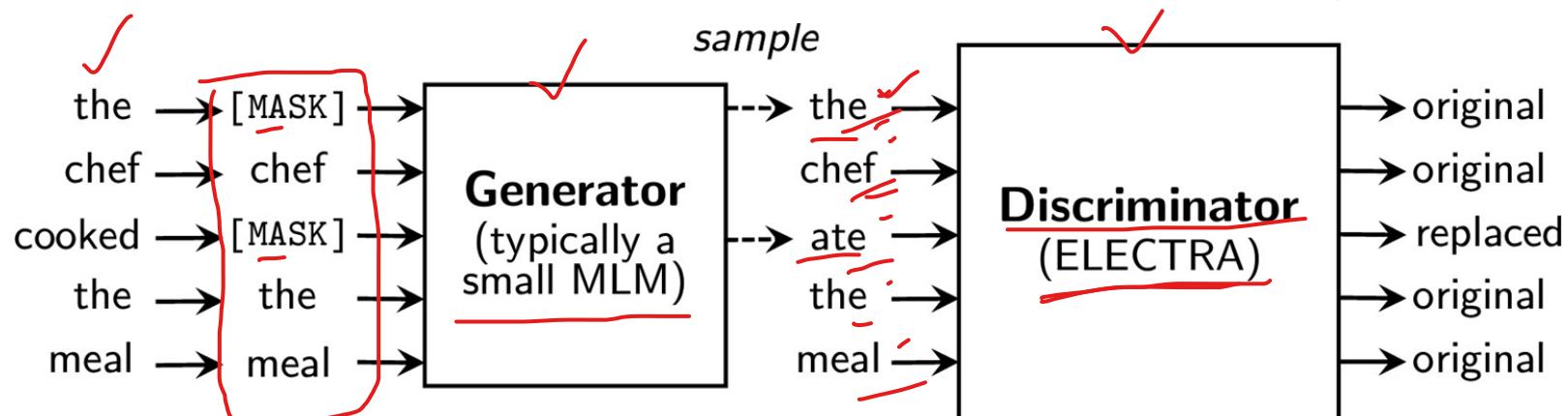
- RoBERTa (RoBERTa: A Robustly Optimized BERT Pretraining Approach. Liu et al. 2019)

- Training the model longer on more data with bigger batches
- Remove the next sentence prediction objective
- Dynamically change the [MASK] patterns in each epoch



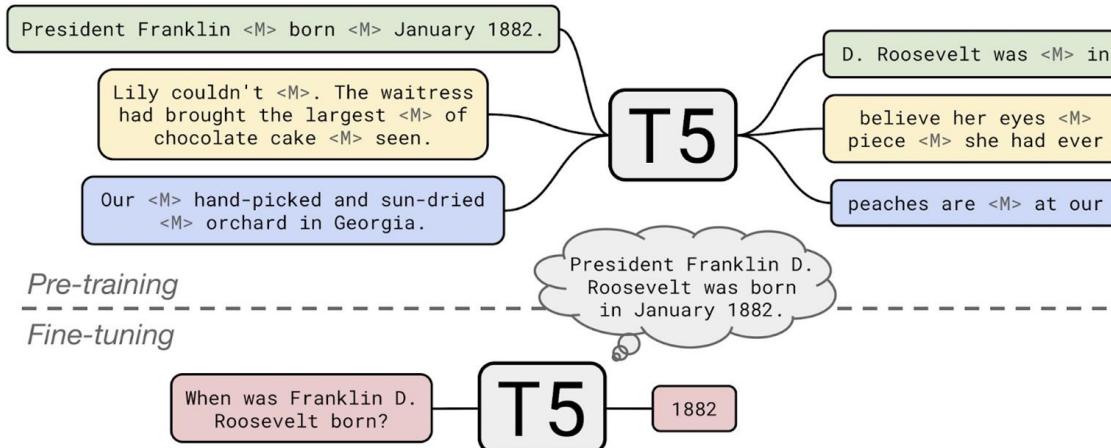
Variants of BERT Model

- ✓ • ELECTRA (ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. Clark et al. 2020)
 - Replaced token detection by corrupting text sequences with an auxiliary MLM
 - Works better than BERT because the input text for ELECTRA does not contain [MASK] tokens (no discrepancy between training and test data)



T5 Model

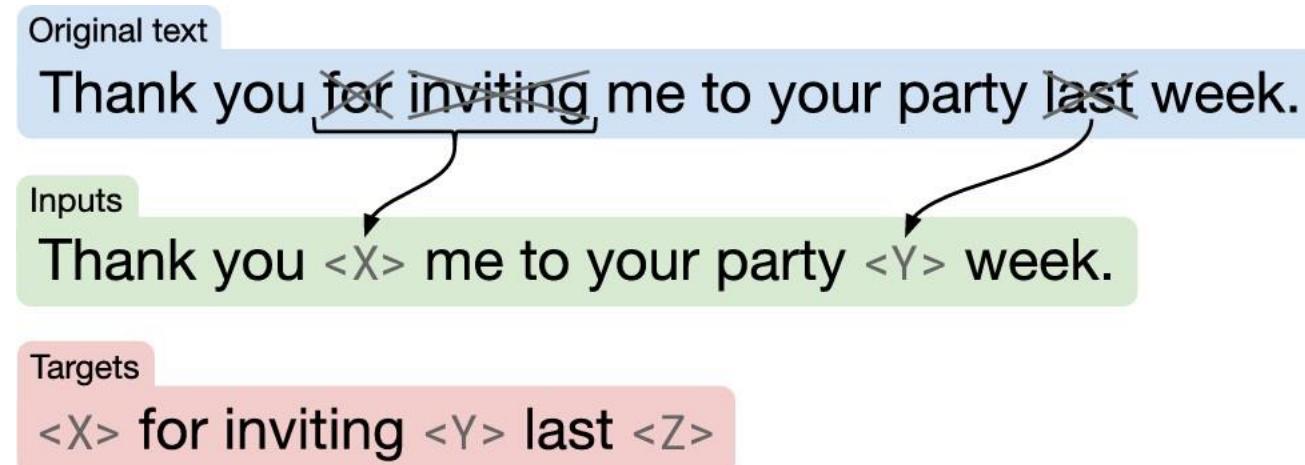
- How to predict a span of masked tokens within a sentence?
- BERT model requires the number of [MASK] token to be given in prior, while GPT models are causal left-to-right models
- T5: Text-to-Text Transfer Transformer (parameters: 60M~11B)



Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Training of T5 Model

- Pretraining: Mask out spans of texts; generate the original spans
- Fine-Tuning: Convert every task into a sequence-to-sequence generation problem



T5 Attention

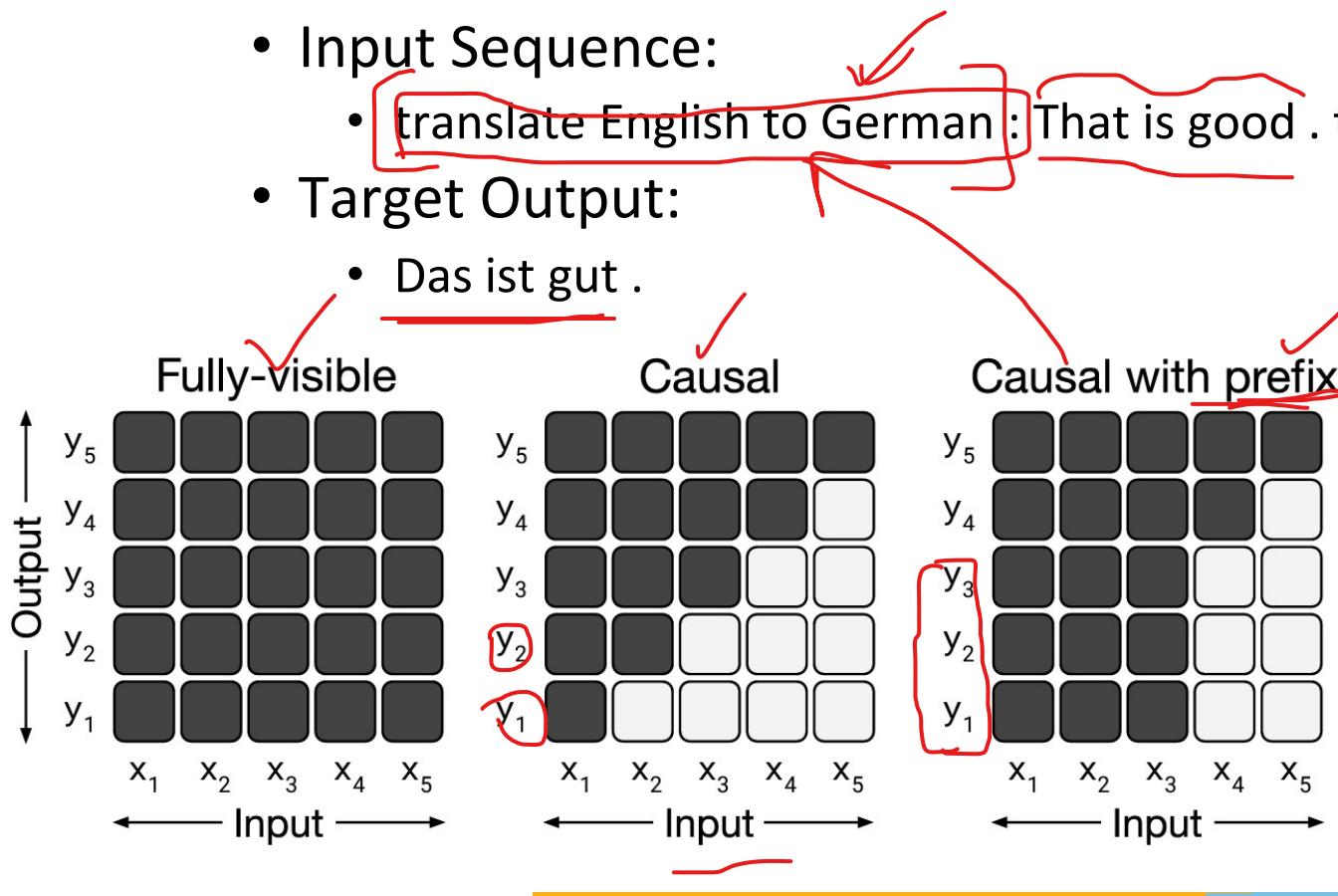
- A “fully-visible” attention mechanism is placed at the input sequence.

- Input Sequence:

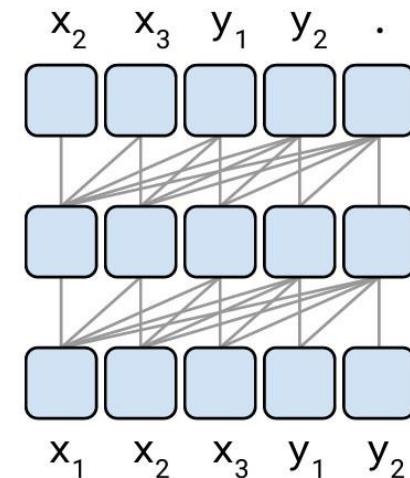
translate English to German : That is good . target :

- Target Output:

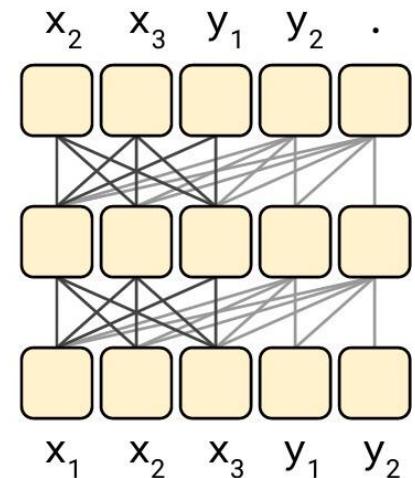
Das ist gut .



Language model

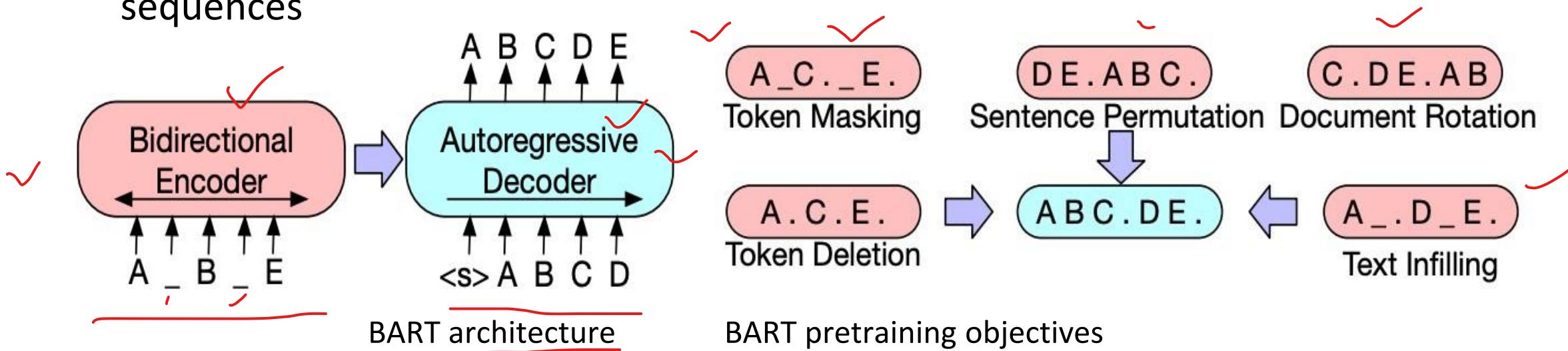


Prefix LM



BART Model

- BART: Denoising autoencoder for pretraining sequence-to-sequence models
- Pretraining: Apply a series of noising schemes (e.g., masks, deletions, permutations...) to input sequences and train the model to recover the original sequences



Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020).

BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ACL.

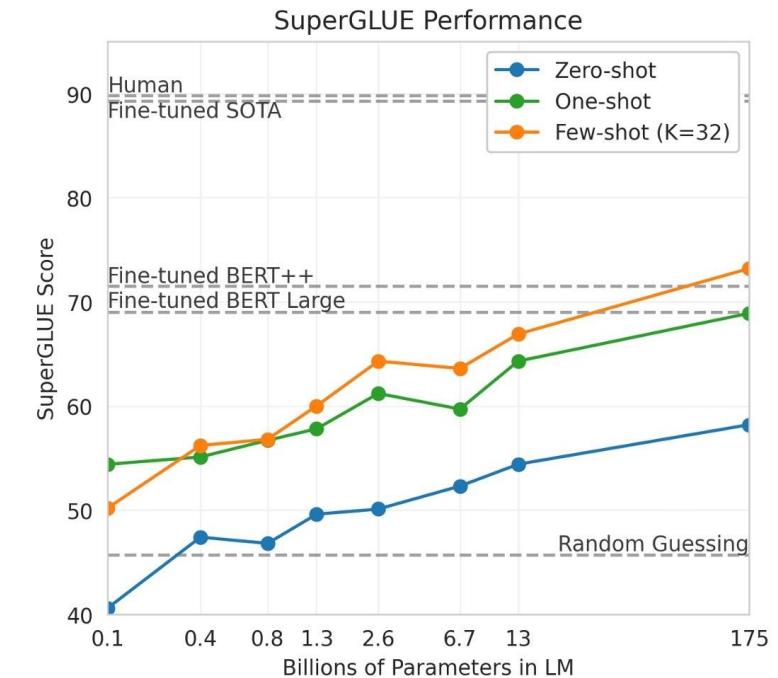
Scaling up Language Models

- GPT-2 model size: 1.5 billion parameters
- Pre-trained model can be very, very large (GPT-3 has 175 billion parameters!) and have very strong text generation abilities.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

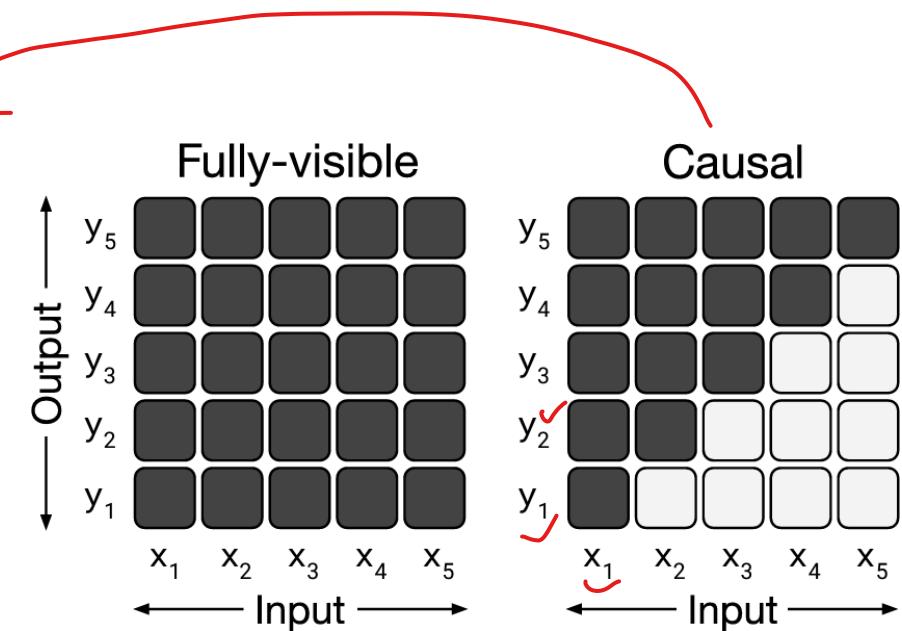
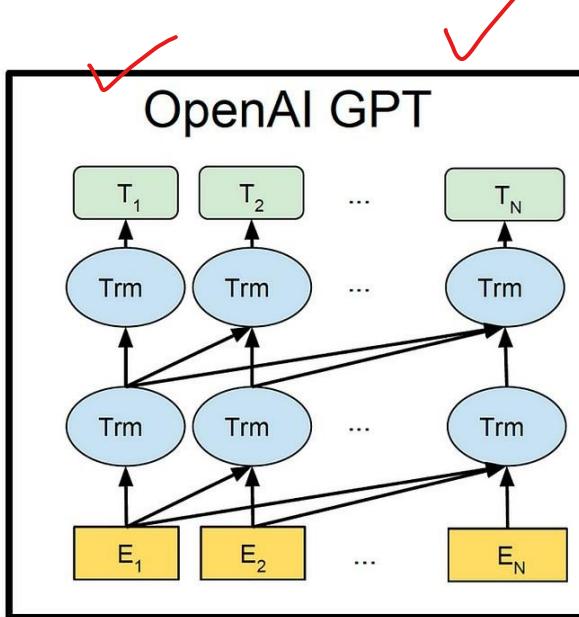
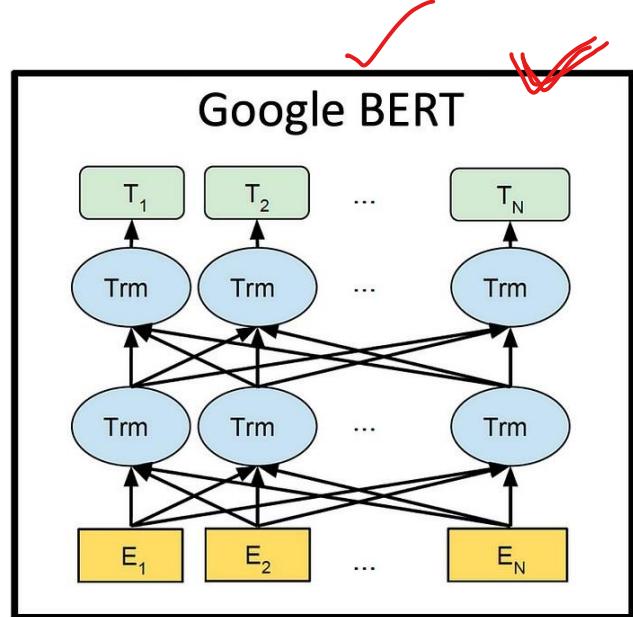
Performance of Zero-Shot/Few-Shot GPT-3

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	89.0	91.0	96.9	93.9	94.8	92.5
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0
	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	76.1	93.8	62.3	88.2	92.5	93.3
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1



Discussion Question

- There has been fewer successful attempts to scaling up BERT-based bidirectional models (e.g., 100x) than unidirectional models.
- Why does unidirectional model have better scaling performance than bidirectional models?



Large Language Model Pre-training Framework

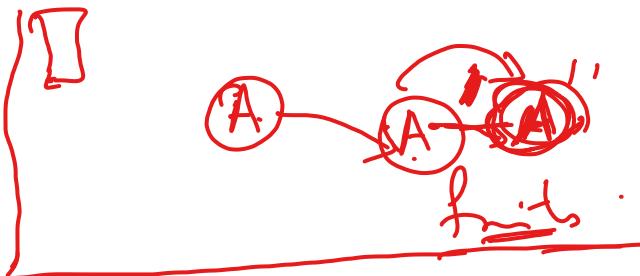
- ChatGPT training procedure

- Self-supervised pre-training

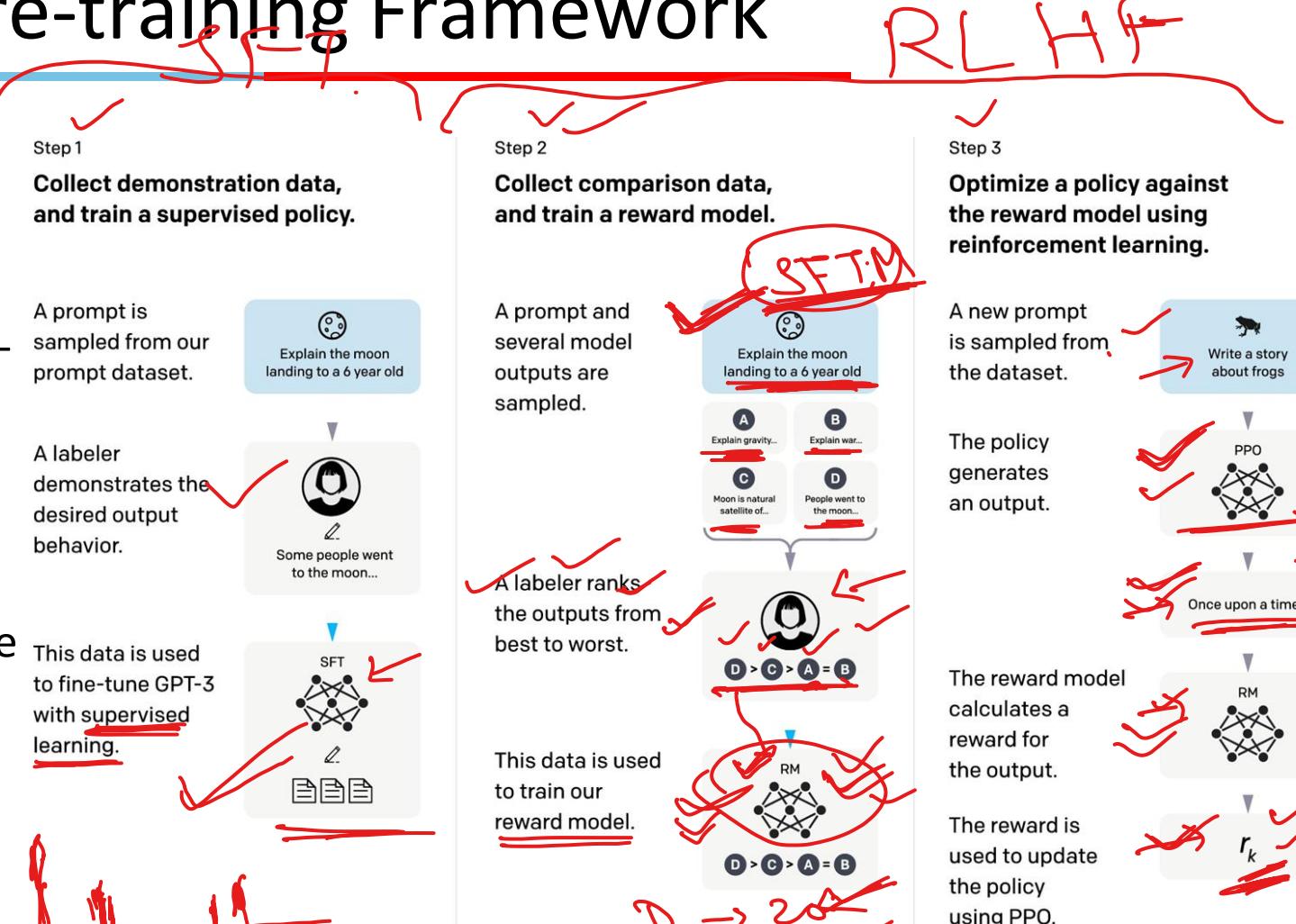
✓ ✓ ~~Supervised training on pairs of human-written data (Step 1)~~

- Model generate multiple outputs for a prompt, train a reward model on human-labeled ranking list (Step 2)

- Optimize the language model with the trained reward model (Step 3)



Training language models to follow instructions with human feedback. Ouyang et al. 2022.



Large Language Model Pre-training Framework

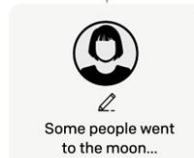
Step 1

Collect demonstration data, and train a supervised policy.

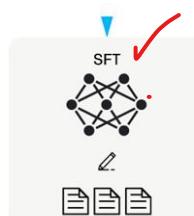
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

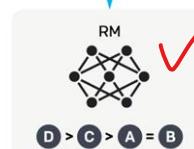
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



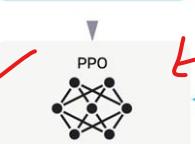
Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

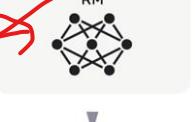


The policy generates an output.



Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

Instruction-Tuning

Reinforcement Learning from Human Feedback

Training language models to follow instructions with human feedback. Ouyang et al. 2022.

LLaMA: Open and Efficient Foundation Language Models

- ✓ • ChatGPT (175B): **closed-source/proprietary models** where we don't know about their pre-training corpus, and don't have access to their pre-trained weights, and can only do inference or training through APIs
- ✓ • LLaMA (7B/13B/33B/65B): **open-source models** where the pre-trained corpus is transparent, and we have full access to their pre-trained weights
 - Pre-trained Corpus: English CommonCrawl, C4, Github, Wikipedia, Gutenberg and Books3, ArXiv, Stack Exchange.
 - Smaller models allow researchers with limited computing resources to understand how and why these language models work

Language Models and User intents

- Language models pre-trained on large corpus not necessarily aligned with user intents

Inference with T5 model

⚡ Inference API ⓘ

Text2Text Generation

Who is Donald Trump?

Compute

⌘+Enter

0.7

Computation time on cpu: 0.578 s

's President? Who is Hillary Clinton? Who is Donald Trump?? Who is

Instruction-tuned T5 model

⚡ Inference API ⓘ

Text2Text Generation

Examples ▾

Who is Donald Trump?

Compute

⌘+Enter

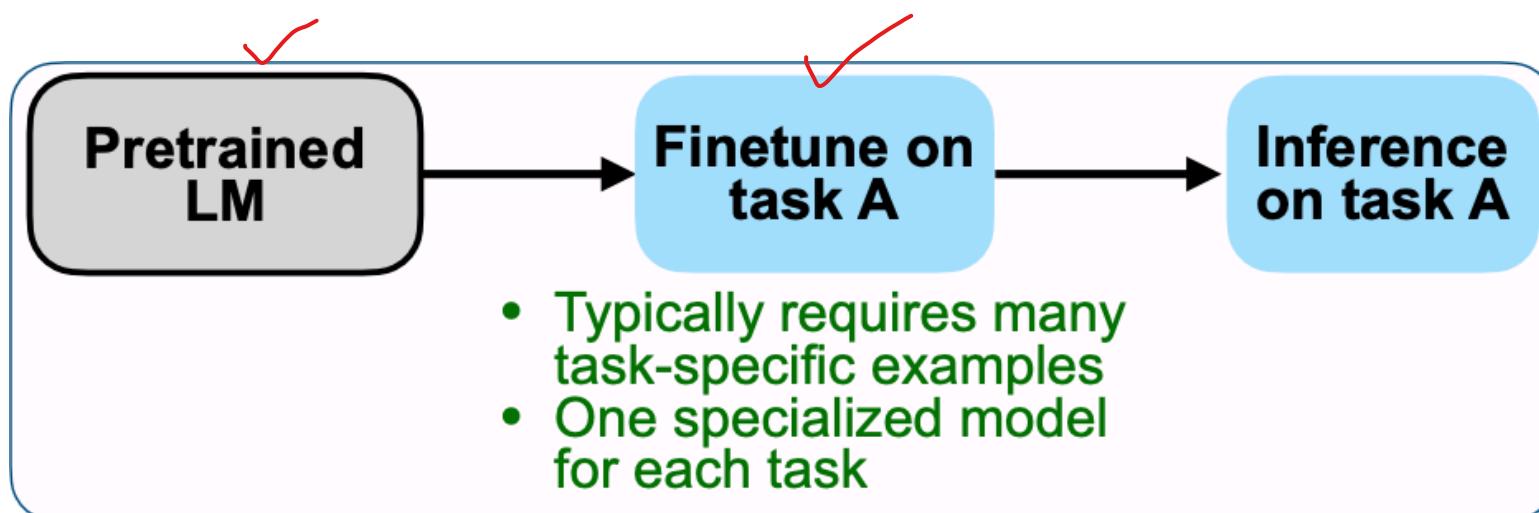
0.2

Computation time on cpu: 0.159 s

president of united states

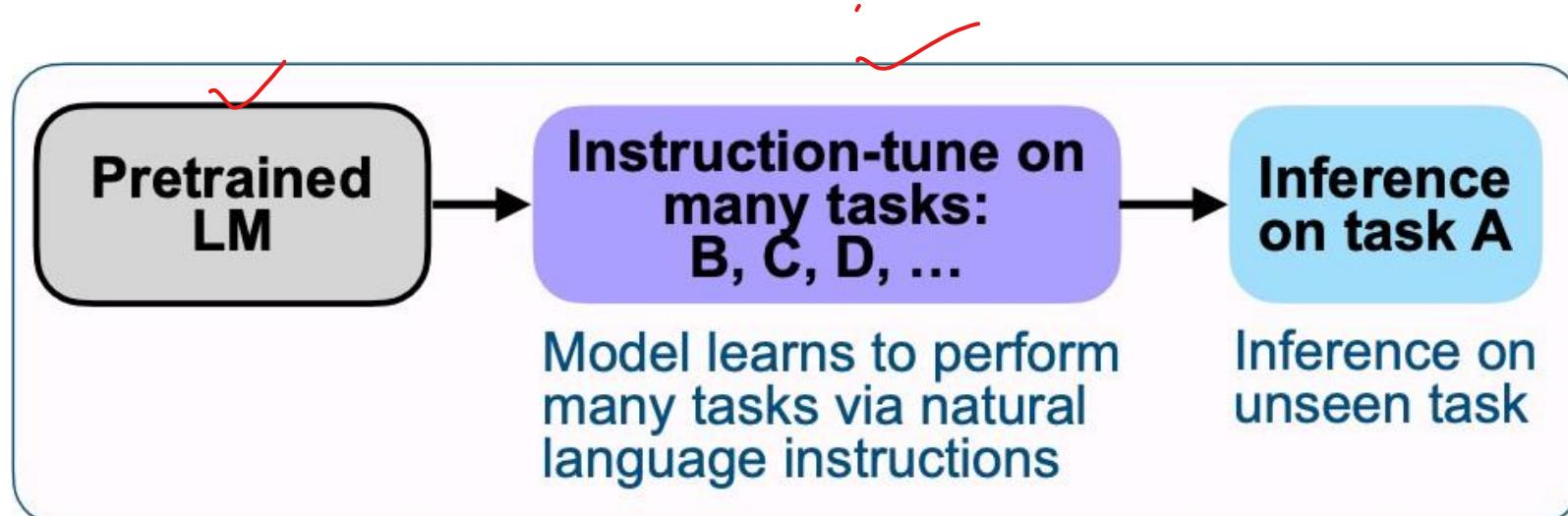
Recall Finetuning

- The pre-training stage let language models learn generic representation and knowledge from the corpus, but they are not specifically fine-tuned on any form of user tasks.
- To adapt language models to a specific downstream task, we could use task-specific datasets for fine-tuning

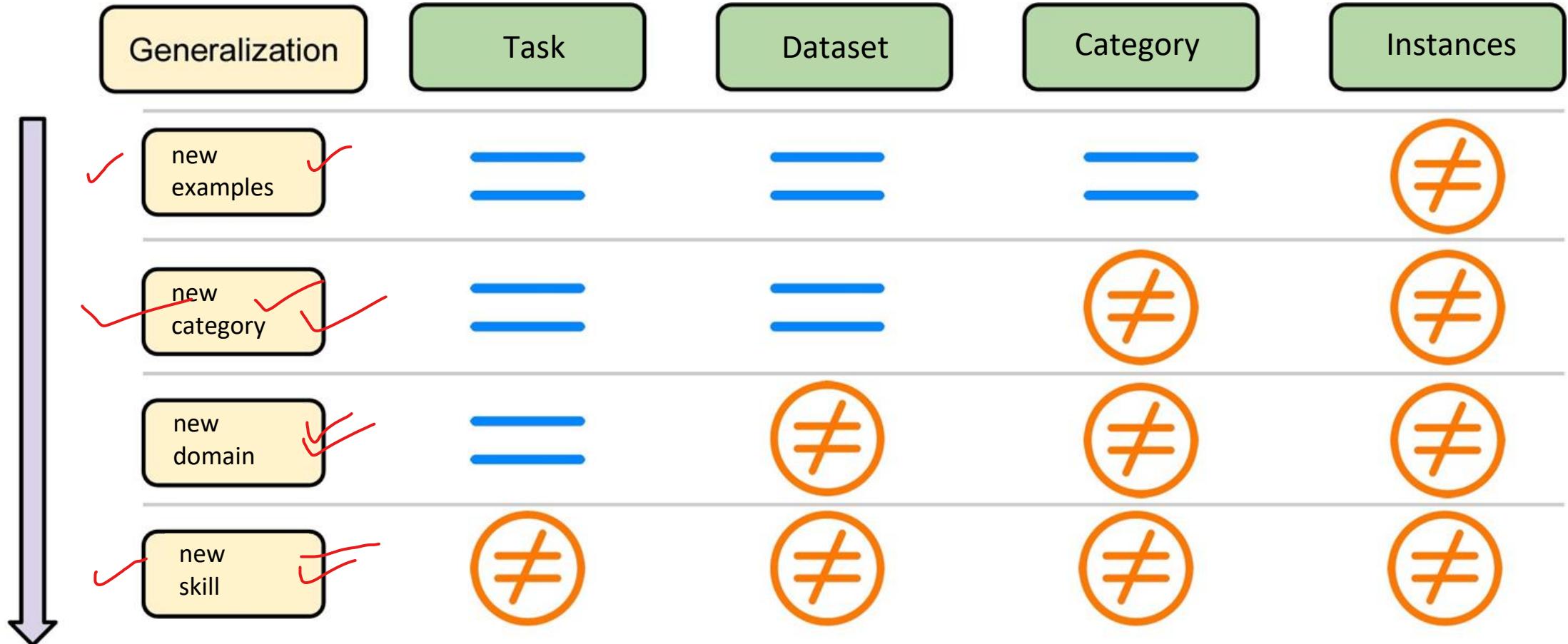


Instruction Tuning

- Fine-tuning on many tasks! Teach language models to follow different natural language instructions, so that it could perform better on downstream tasks and generalize to unseen tasks!
- Fine-tuning -> Instruction Pre-training



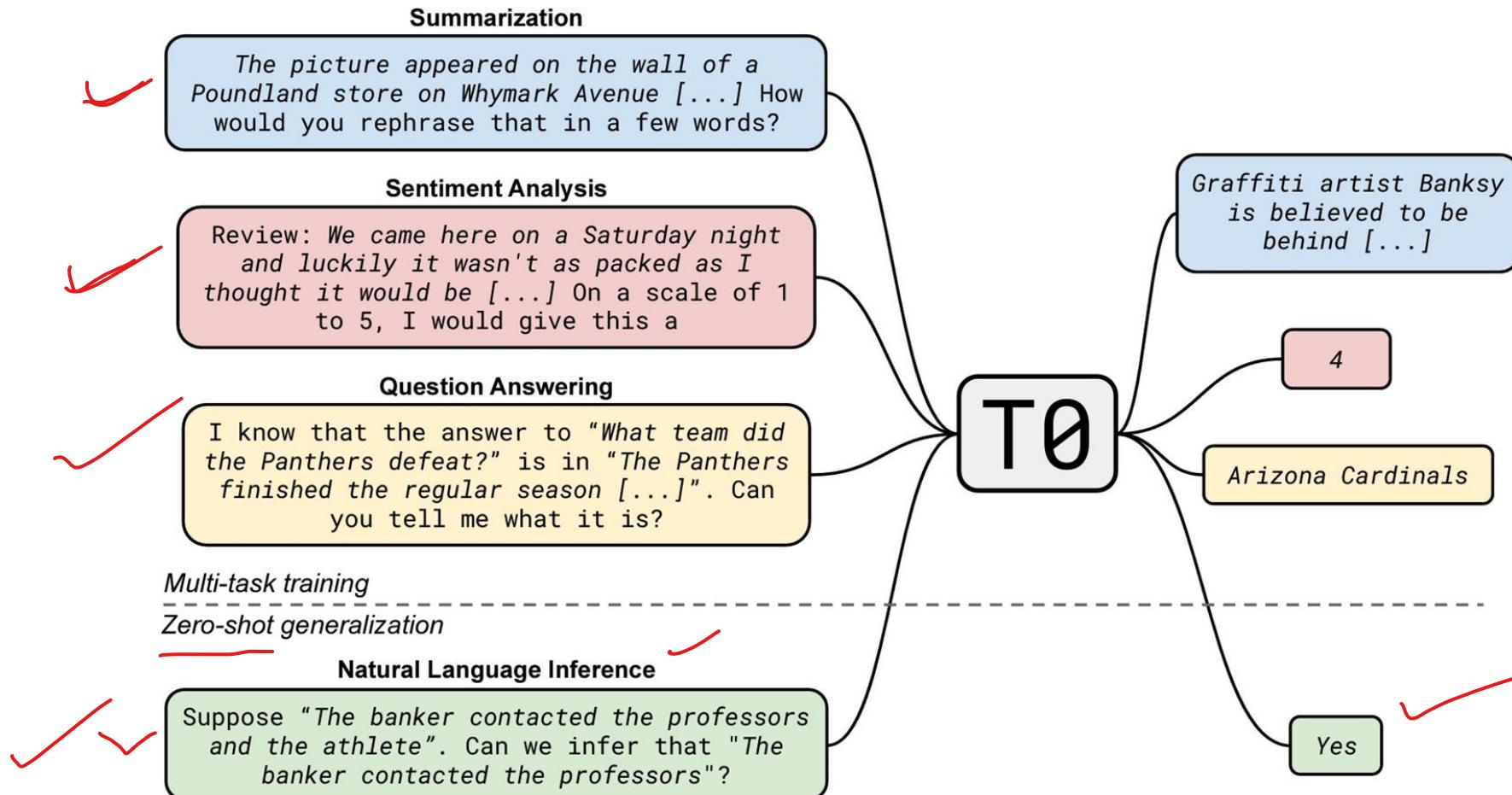
Increasing Generalization



Increasing
generalization

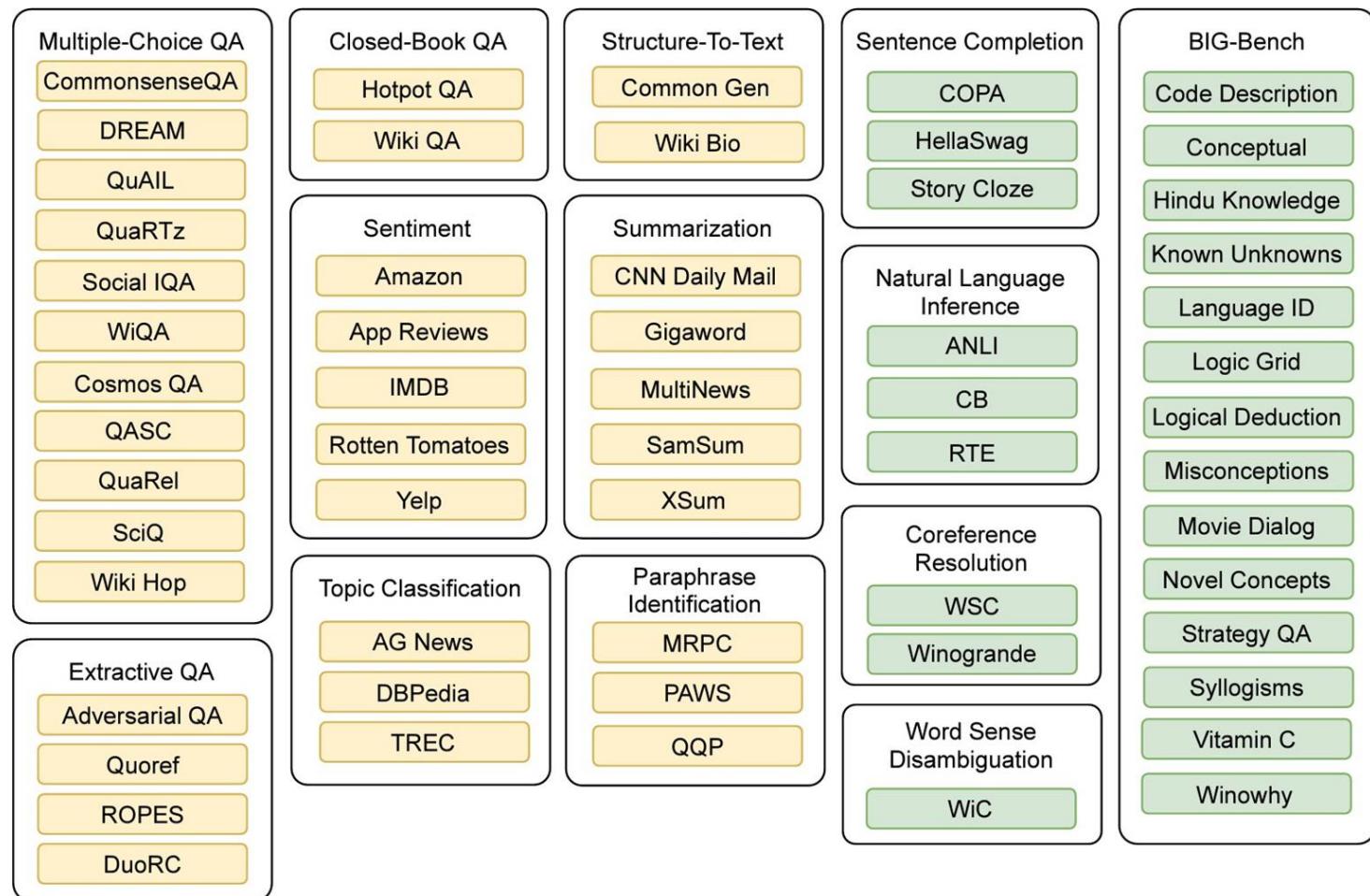
Multitask Prompted Training Enables Zero-Shot Task Generalization

(Sanh et. al, 2021)



T0 Training Datasets

- Collecting from multiple public NLP datasets



Training Mixtures and Held-Out Sets

- ✓ • Training mixtures:
 - ✓ • QA (Question Answering tasks), structure-to-text, summarization
 - ✓ • Sentiment analysis, topic classification, paraphrase identification
- ✓ • Held-out test set:
 - Sentence completion, BIG-Bench
 - Natural language inference, coreference resolution, word sense disambiguation

Task Adaptation with Prompt Templates

- Instead of directly using pairs of input and output, add specific instructions to explain each task (different templates per task)
- the outputs are tokens instead of class labels

QQP (Paraphrase)

Question1	How is air traffic controlled?
Question2	How do you become an air traffic controller?
Label	0

{Question1} {Question2}
Pick one: These questions
are duplicates or not
duplicates.

I received the questions
"{Question1}" and
"{Question2}". Are they
duplicates?

{Choices[label]}

{Choices[label]}

XSum (Summary)

Document	The picture appeared on the wall of a Poundland store on Whymark Avenue...
Summary	Graffiti artist Banksy is believed to be behind...

{Document}
How would you
rephrase that in
a few words?

{Summary}

First, please read the article:
{Document}
Now, can you write me an
extremely short abstract for it?

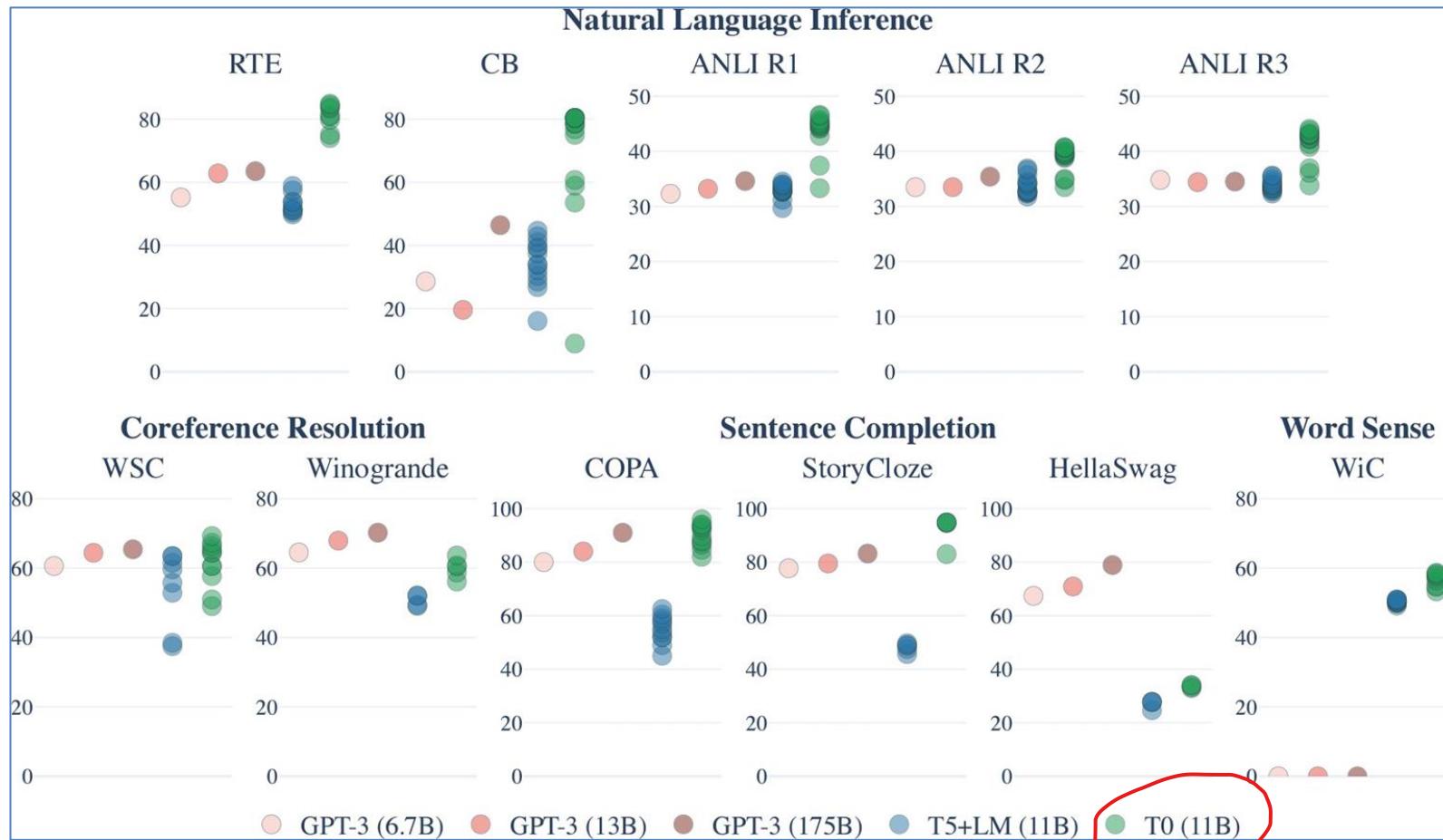
{Summary}

Experiments

- The proposed model is called T0, trained on T5-LM (11B model) with multitask mixture of training sets
- Baselines

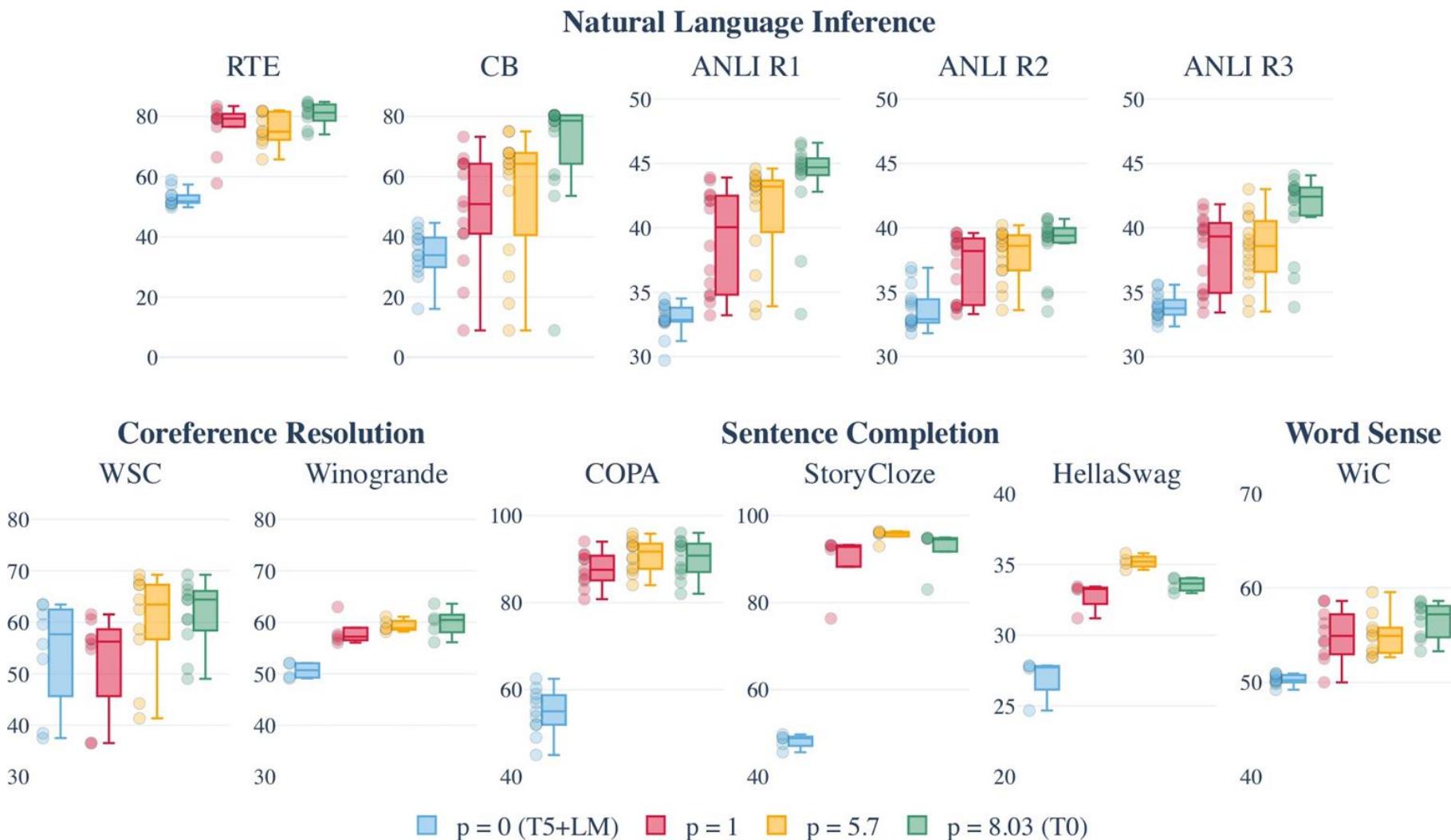
(light pink) GPT-3 (6.7B) (pink) GPT-3 (13B) (brown) GPT-3 (175B) (blue) T5+LM (11B)

Performance on Held-Out Tasks

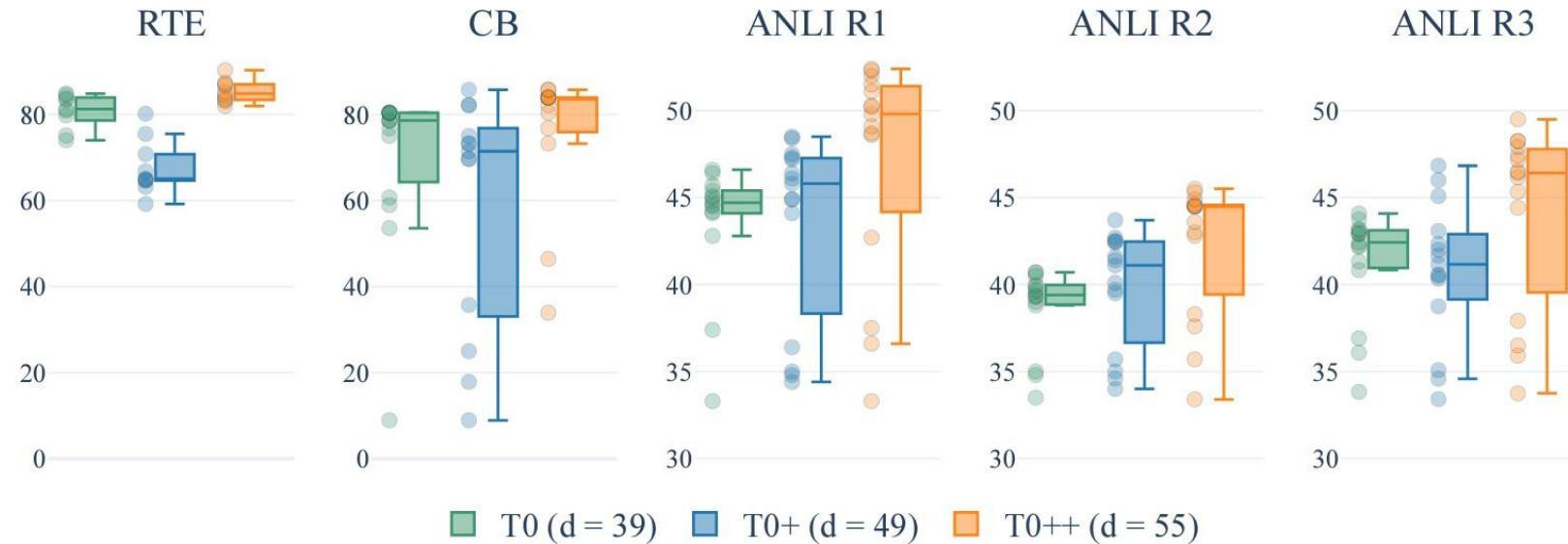


- For T5 and T0 models, each dot represents one evaluation prompt.

Effects of Prompt Numbers



Effects of More Training Datasets

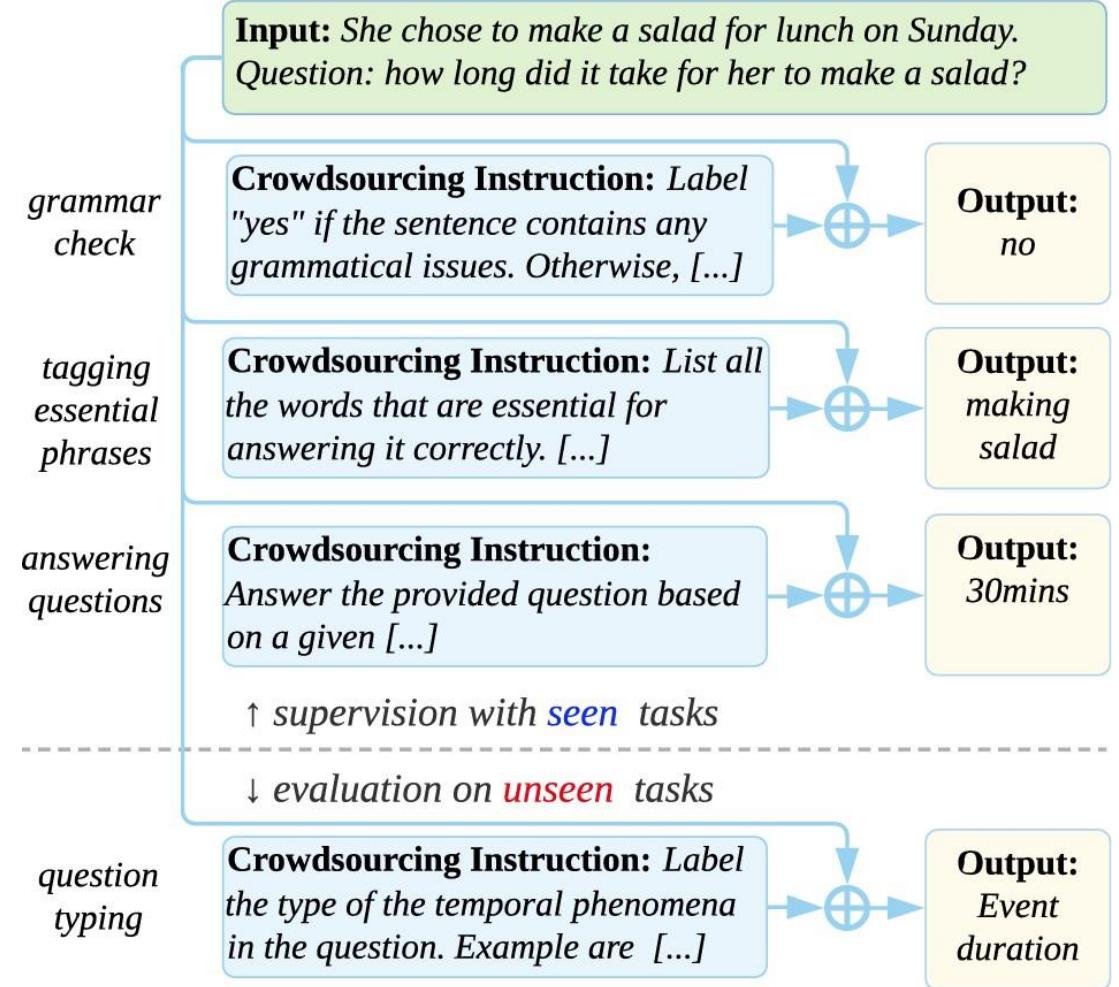


- Adding more datasets consistently leads to higher median performance

Cross-Task Generalization via Natural Language Crowdsourcing Instructions

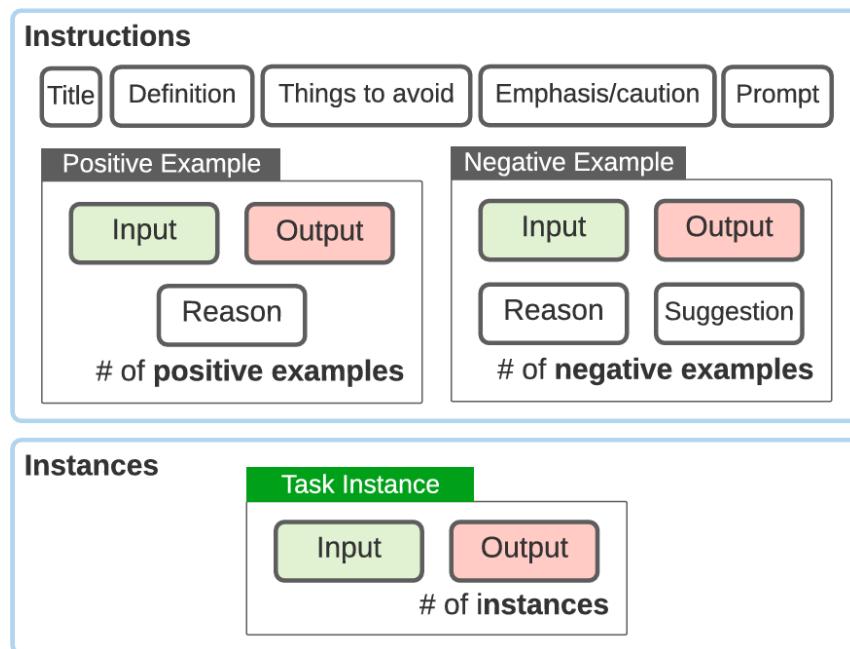
(Mishra et. al, 2021)

- ✓ • Contribution:
- ✓ • A crowdsourced dataset: Natural Instructions
 - 61 distinct tasks
 - 193k instances (input -> output)
- ✓ • A more complete instruction schema



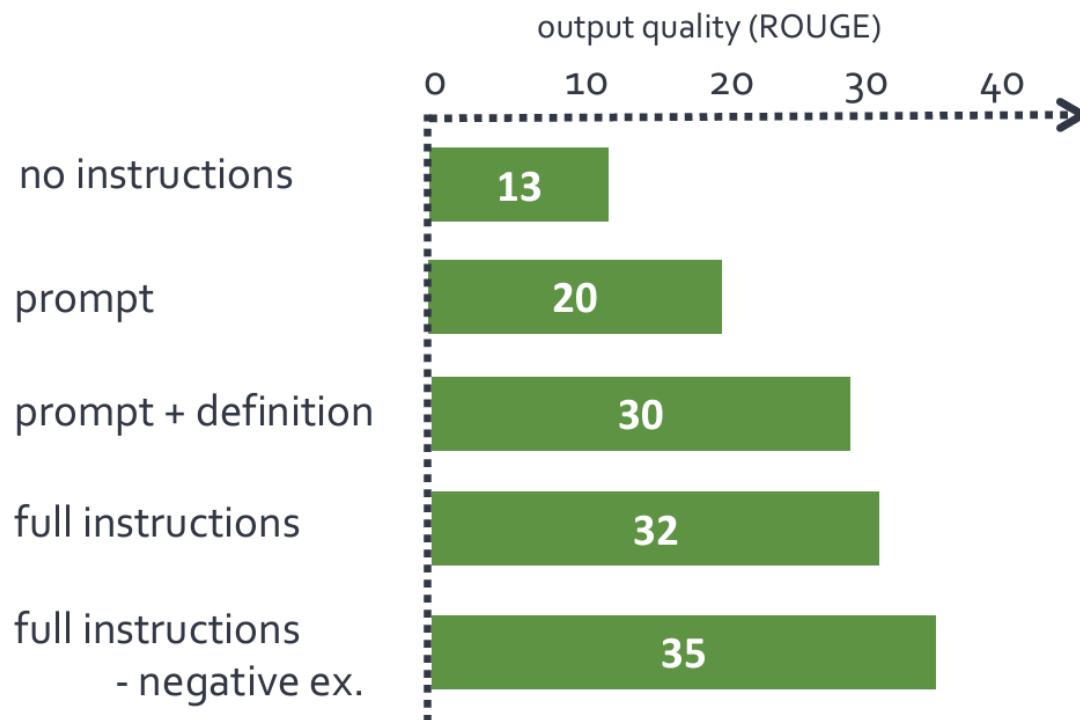
Crowdsourced Dataset

- 1. Randomly split the tasks (12 evaluation tasks, 49 supervision tasks)
- 2. Leave-one-category-out



Experiments: Generalization to Unseen Tasks

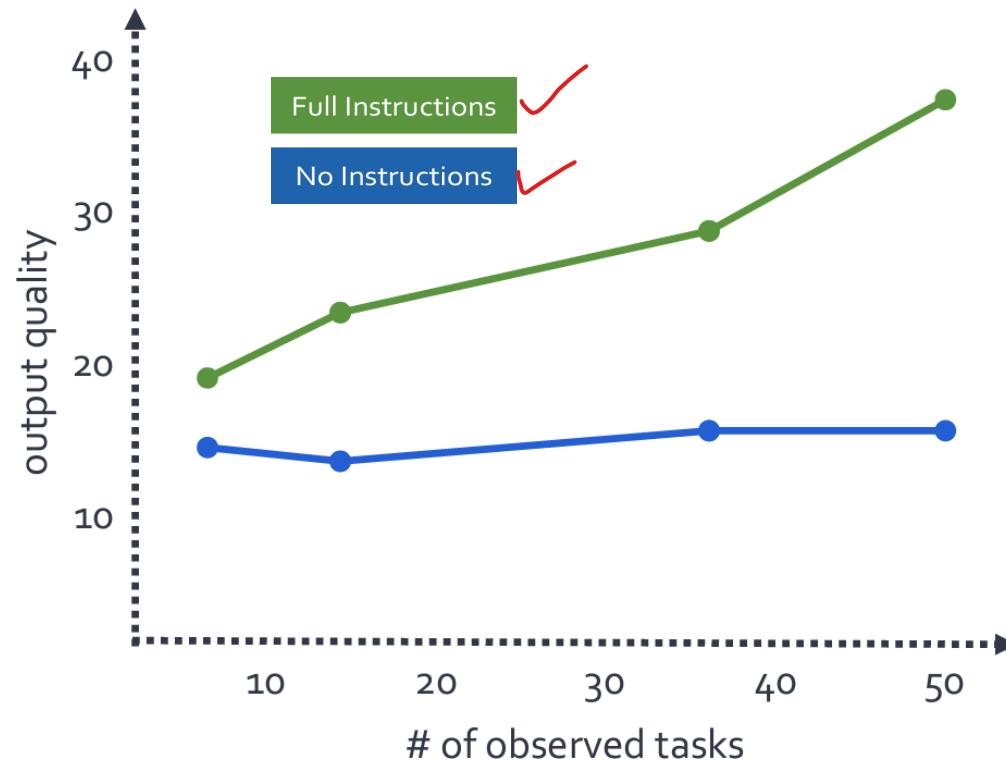
- Model: BART (140M params., instruction-tuned)



- All instruction elements (except negative examples) help improve model performance on unseen tasks.

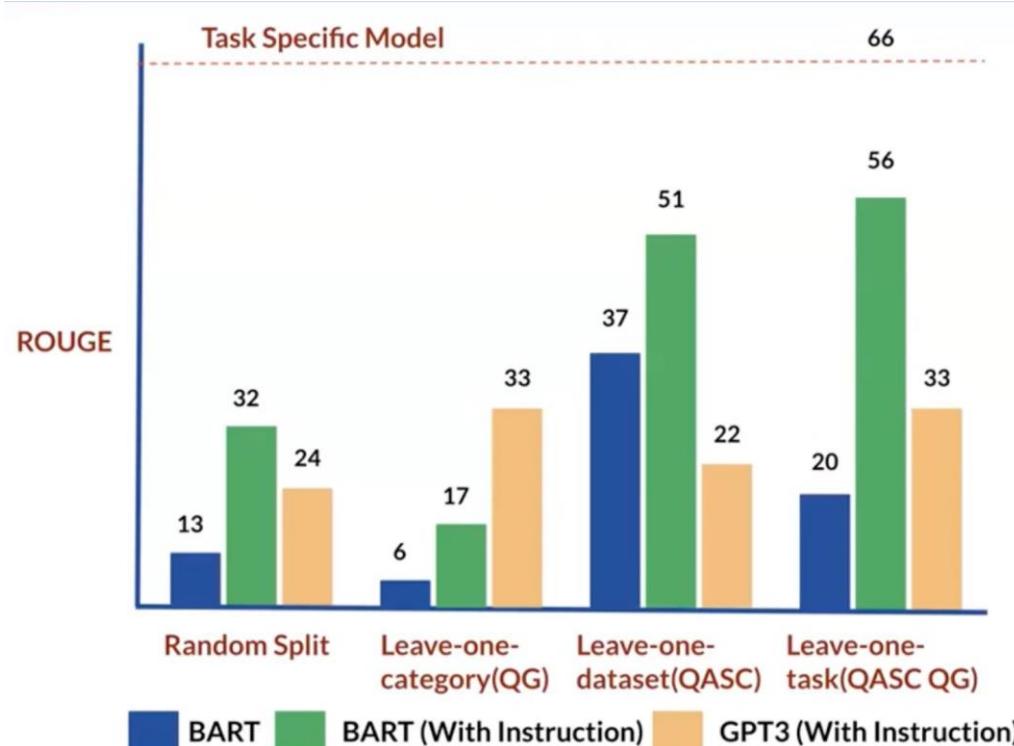
Experiments: Number of Training Tasks

Generalization to unseen tasks improves with more observed tasks



Experiments: Comparison with Larger Model

- Model: BART (140M params., instruction-tuned)
- Baseline: GPT3 (175B params., not instruction-tuned)



- Instructions consistently improve model performance on unseen tasks.
- When both having access to instructions on the held-out set, BART (instruction-tuned) can often outperform GPT3 (not instruction-tuned)

Self-Instruct: Aligning Language Models with Self- Generated Instructions

(Wang et. al, 2022)

- ✓ • Human-written instruction data can be very expensive!
- Can we reduce the human annotations?
- Idea: bootstrap from off-the-shelf LMs

Self-Instruct: Aligning Language Models with Self- Generated Instructions

(Wang et. al, 2022)

- Human written seed tasks to bootstrap off-the-shelf language models (GPT-3)

- I am planning a 7-day trip to Seattle. Can you make a detailed plan for me?
- Is there anything I can eat for breakfast that doesn't include eggs, yet includes protein and has roughly 700-1000 calories?
- Given a set of numbers find all possible subsets that sum to a given number.
- Give me a phrase that I can use to express I am very happy.



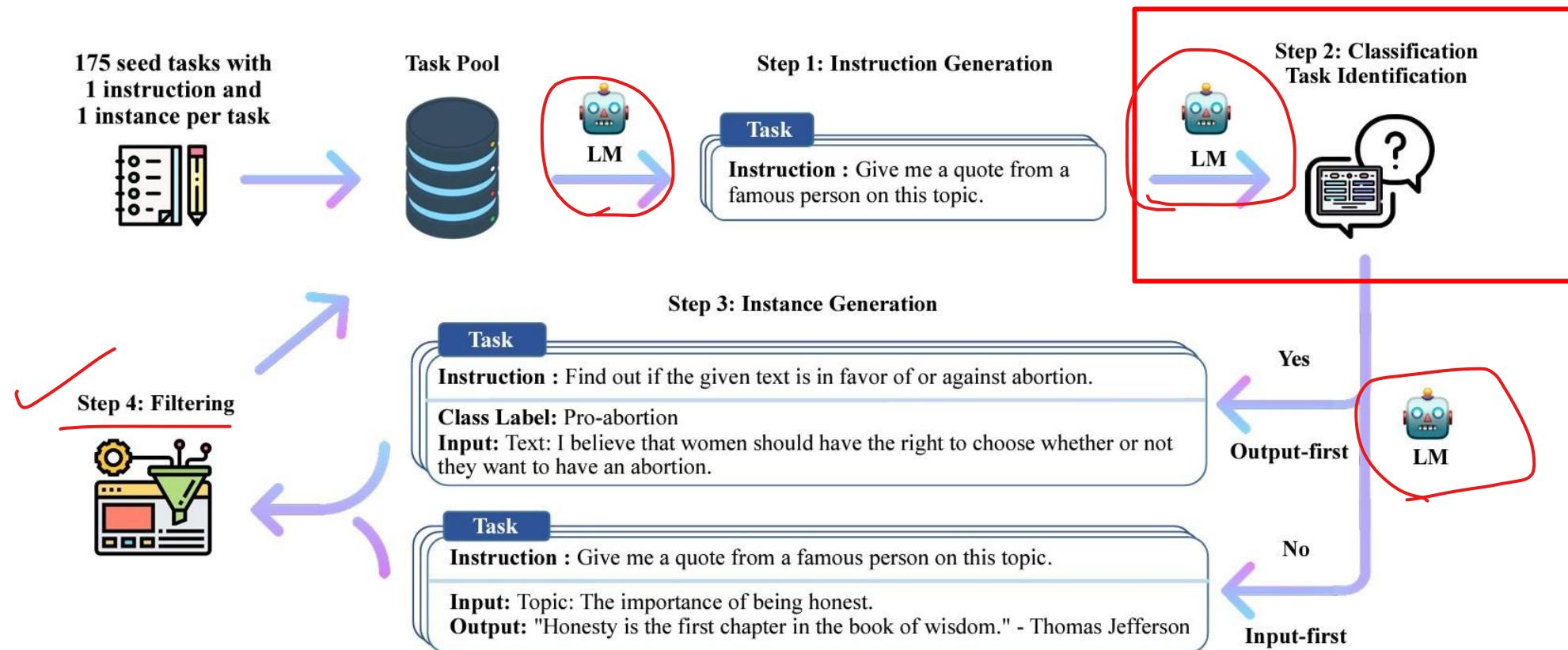
LM

Pre-trained, but **not aligned yet**

- Create a list of 10 African countries and their capital city?
- Looking for a job, but it's difficult for me to find one. Can you help me?
- Write a Python program that tells if a given string contains anagrams.

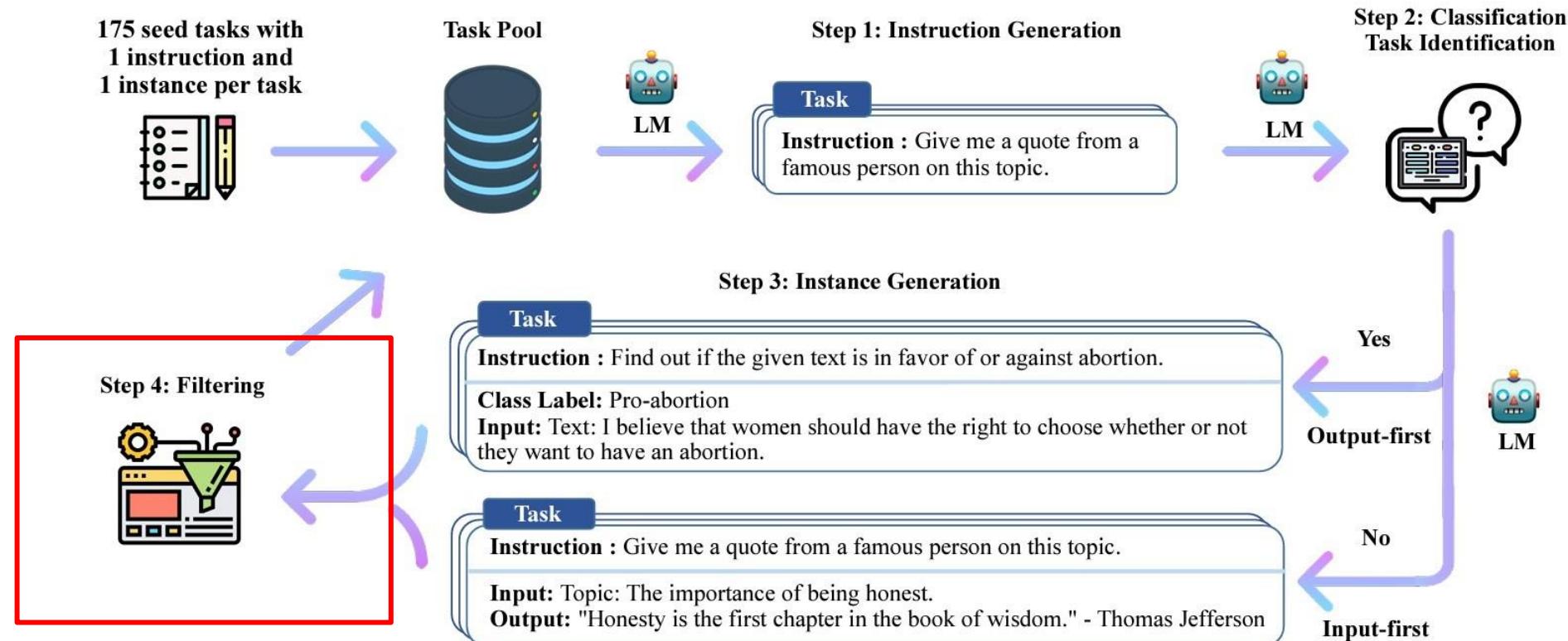
Self-Instruct Framework

- Classify whether the generated instruction is a classification task
- Output-first: avoid bias towards one class label



Self-Instruct Framework

- Filter out instructions similar with existing ones
- Add newly generated tasks into the task pool for next iteration



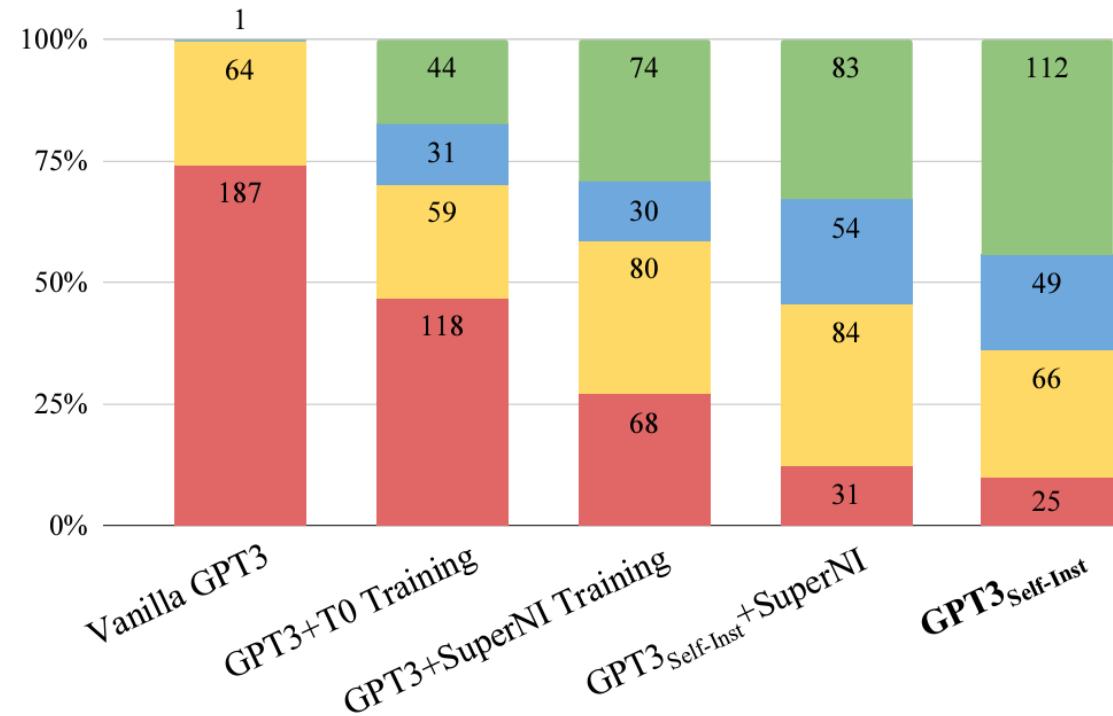
Experiment Results

- Use a GPT-3 (“davinci”) model to generate new instruction tasks, and fine-tune the GPT-3 model itself
- 175 seed tasks -> 52K instructions and 82K instances

statistic	
# of instructions	52,445
- # of classification instructions	11,584
- # of non-classification instructions	40,861
# of instances	82,439
- # of instances with empty input	35,878
ave. instruction length (in words)	15.9
ave. non-empty input length (in words)	12.7
ave. output length (in words)	18.9

Evaluation on User-Oriented Instructions

■ A: correct and satisfying response ■ B: acceptable response with minor imperfections
■ C: responds to the instruction but has significant errors ■ D: irrelevant or invalid response



- Self-training the model by bootstrapping instruction tasks from limited human-written seed tasks can improve model alignment

LIMA: Less Is More for Alignment

(Zhou et. al, 2023)

- Can we use a small number of data to generalize to new tasks?
- Hypothesis: A model's knowledge and capabilities are learned almost entirely during pre-training, while alignment teaches it the right format to be used when interacting with users

LIMA: Less Is More for Alignment (Zhou et. al, 2023)

- 1000 training examples: no more distillation data and with minor human annotations (200)
 - ✓ 750 top questions selected from community forums
 - ✓ manually write 250 examples of prompts and responses to emphasize the response style of an AI assistant
- Finally train a 65B LLaMa model on 1000 demonstrations.

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334
Dev			
Paper Authors (Group A)	50	36	N/A
Test			
Pushshift r/AskReddit	70	30	N/A
Paper Authors (Group B)	230	31	N/A

LIMA: Less Is More for Alignment

- Quality and diversity are the keys
- Quality Control:
 - Public data: select data with higher user ratings
 - In-house authored data: uniform tone and format
- Diversity Control:
 - Public data: Stratified sampling to increase domain diversity
 - In-house authored data: Increase task/scenario

Comparing LIMA with other LLMs

- Ask human crowd workers and GPT-4 which model response is better

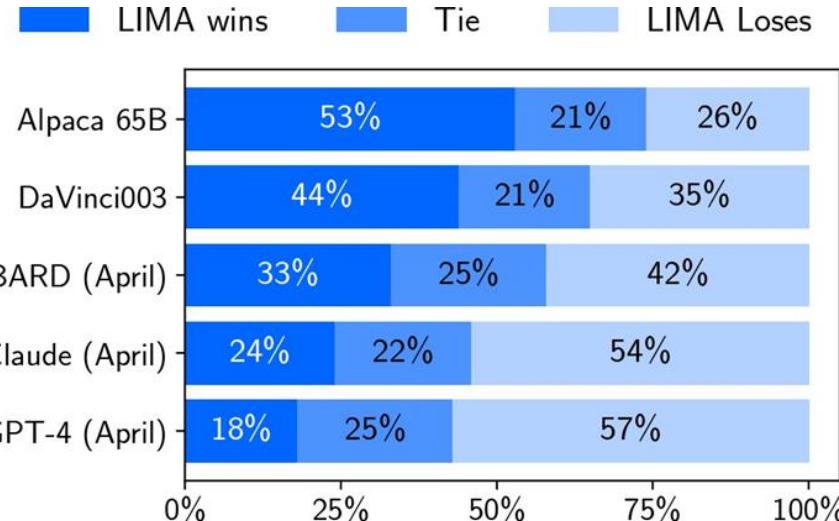


Figure 1: Human preference evaluation, comparing LIMA to 5 different baselines across 300 test prompts.

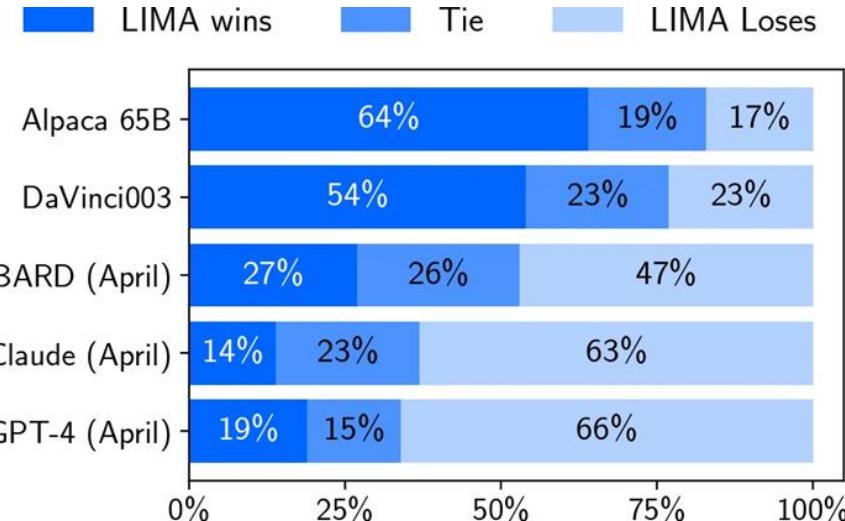


Figure 2: Preference evaluation using GPT-4 as the annotator, given the same instructions provided to humans.

Quality vs. Quantity vs. Diversity

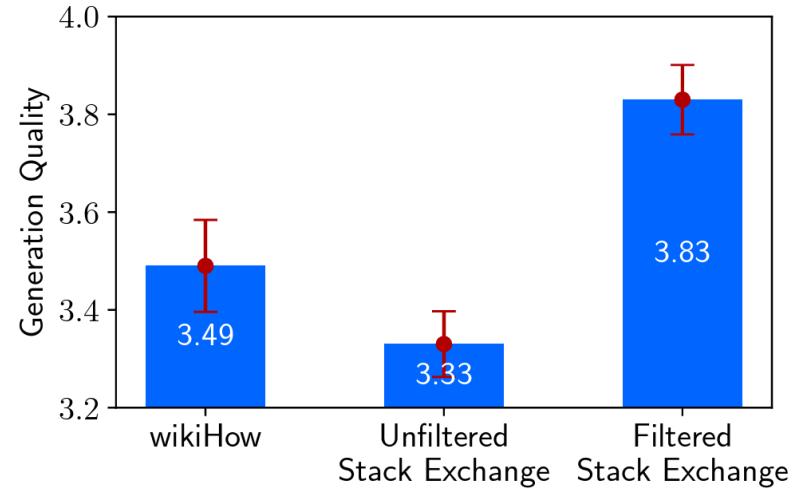
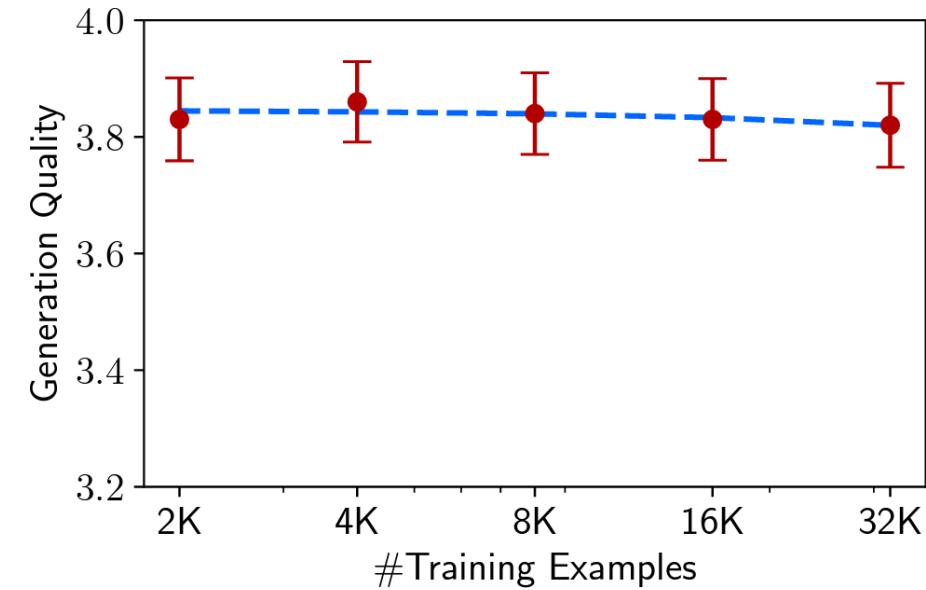


Figure 5: Performance of 7B models trained with 2,000 examples from different sources. **Filtered Stack Exchange** contains diverse prompts and high quality responses; **Unfiltered Stack Exchange** is diverse, but does not have any quality filters; **wikiHow** has high quality responses, but all of its prompts are “how to” questions.

- Scaling up training data does not necessarily improve the model response quality



Quality of Long-Form Highly Structured Response

- LIMA w. or w/o 6 format constraint examples
 - generating a product page with *highlights, about the product* and *how to use*
 - paper reviews with *summary, strengths, weakness* and *potentials*

LIMA w/o 6 Format Constraint Examples (994 Examples)	LIMA (1,000 Examples)
<p>[Marketing Plan (Out of Distribution)]</p> <p>I want to start a local coffee shop. Create a marketing plan with the following elements: Marketing Goals and Objectives, Define Target Audience, Research Marketing Tactics, Plan Marketing Tactics and Develop Your Timeline and Budget.</p>	<p>Here's a marketing plan for your local coffee shop.</p> <p>Marketing Goals and Objectives</p> <ul style="list-style-type: none"> • Increase awareness of our brand among consumers [...] • Drive foot traffic [...] during slow business hours [...] • Build relationships with influencers in our community who [...] <p>Define Target Audience</p> <p>Our target audiences include:</p> <ul style="list-style-type: none"> • Residents of the neighborhood surrounding our coffee shop • Office workers near our coffee shop • Students at nearby schools <p>We will focus our marketing efforts on these audiences because they are most likely to become regular customers at our [...]</p> <p>Research Marketing Tactics</p> <p>Here are some marketing tactics that we could use to reach to reach</p>

Coffee Shop Marketing Plan

Executive Summary

The following marketing plan for Java Jolt Coffee Bar outlines an aggressive approach to establishing a thriving coffee shop business in downtown Pittsburgh. Java Jolt will capitalize on the popularity of specialty coffees and the growing demand for WiFi hot spots by offering a comfortable environment where customers can enjoy premium coffee drinks while taking advantage of free Internet access.

Market research indicates that there is a significant opportunity for a new coffee bar located in the heart of the downtown business district. Market trends suggest that consumers are willing to pay a higher price for quality specialty coffees served in a welcoming atmosphere. Additionally, many office workers and students living in the area do not have easy access to high-speed Internet service, presenting an additional revenue stream for the coffee bar.

This marketing plan includes the following elements:

Here's a marketing plan for your local coffee shop.

Marketing Goals and Objectives

- Increase awareness of our brand among consumers [...]
- Drive foot traffic [...] during slow business hours [...]
- Build relationships with influencers in our community who [...]

Define Target Audience

Our target audiences include:

- Residents of the neighborhood surrounding our coffee shop
- Office workers near our coffee shop
- Students at nearby schools

We will focus our marketing efforts on these audiences because they are most likely to become regular customers at our [...]

Research Marketing Tactics

Here are some marketing tactics that we could use to reach to reach

Tulu: How Far Can Camels Go?

Exploring the State of Instruction Tuning on Open Resources (Wang et. al, 2023)

- A comprehensive study on different instruction-tuning datasets
- Two mixtures of datasets
 - Human data mixture
 - Human + GPT data mixture

Comparison of Using Different Instruction Tuning Datasets

- There is not a single best instruction tuning dataset across all tasks
- Combining datasets results in the best overall performance

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
Vanilla LLaMa 13B	42.3	14.5	39.3	43.2	28.6	-	-
+SuperNI	49.7	4.0	4.5	50.2	12.9	4.2	20.9
+CoT	44.2	40.0	41.9	47.8	23.7	6.0	33.9
+Flan V2	50.6	20.0	40.8	47.2	16.8	3.2	29.8
+Dolly	45.6	18.0	28.4	46.5	31.0	13.7	30.5
+Open Assistant 1	43.3	15.0	39.6	33.4	31.9	58.1	36.9
+Self-instruct	30.4	11.0	30.7	41.3	12.5	5.0	21.8
+Unnatural Instructions	46.4	8.0	33.7	40.9	23.9	8.4	26.9
+Alpaca	45.0	9.5	36.6	31.1	29.9	21.9	29.0
+Code-Alpaca	42.5	13.5	35.6	38.9	34.2	15.8	30.1
+GPT4-Alpaca	46.9	16.5	38.8	23.5	36.6	63.1	37.6
+Baize	43.7	10.0	38.7	33.6	28.7	21.9	29.4
+ShareGPT	49.3	27.0	40.4	30.5	34.1	70.5	42.0
+Human data mix.	50.2	38.5	39.6	47.0	25.0	35.0	39.2
+Human+GPT data mix.	49.3	40.5	43.3	45.6	35.9	56.5	45.2

Different Base Models

- Base model quality is extremely important for downstream performance
- LLaMA is pre-trained on more tokens than other models

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
Pythia 6.9B	34.8	16.0	29.2	32.8	20.9	23.5	26.2
OPT 6.7B	32.6	13.5	27.9	24.1	8.9	25.9	22.2
LLAMA 7B	44.8	25.0	38.5	43.5	29.1	48.6	38.3
LLAMA-2 7B	49.2	37.0	44.2	52.8	33.9	57.3	45.7

Different Model Sizes

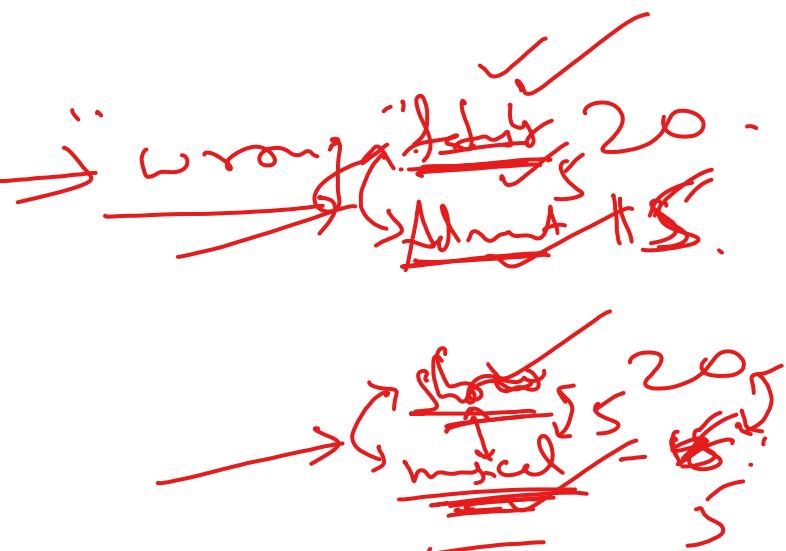
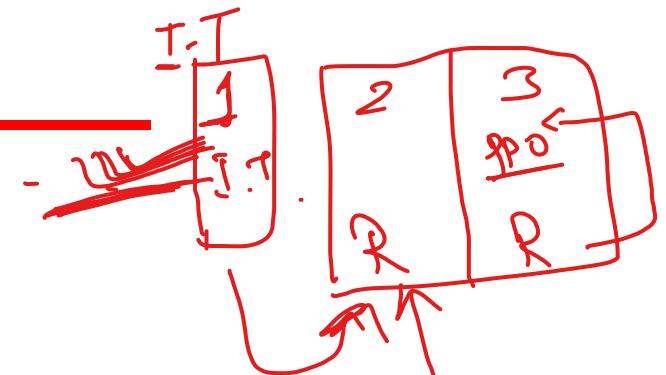
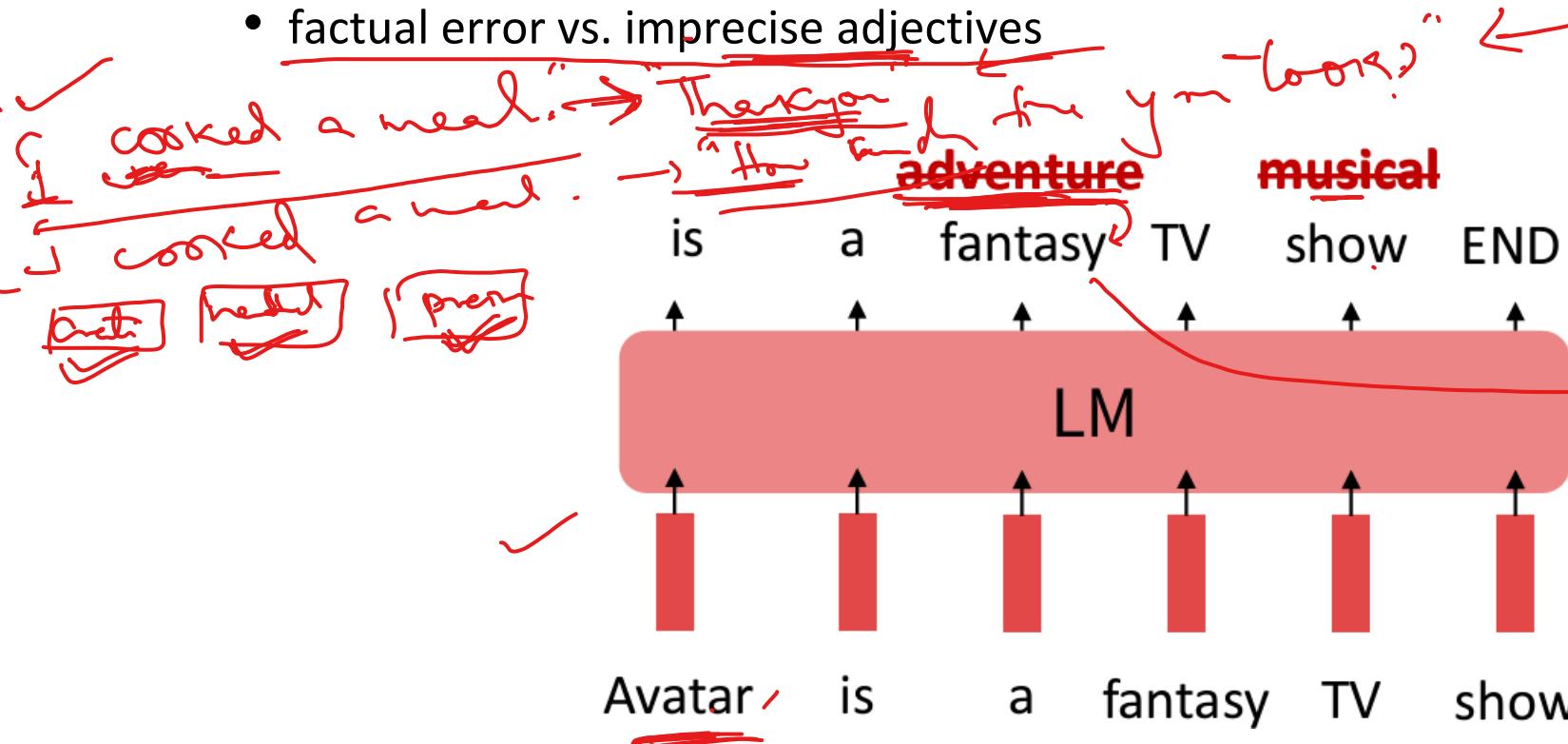
- Smaller models benefit more from instruction-tuning
- Instruction-tuning does not help to enhance strong capabilities already exist in the original model

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
.camel models trained on our final Human+GPT data mixture ↓							
TÜLU 🐫 7B	44.8 (+13.3)	25.0 (+15.0)	38.5 (+5.5)	43.5 (+5.1)	29.1 (+8.6)	48.6	38.3
TÜLU 🐫 13B	49.3 (+7.0)	40.5 (+26.0)	43.3 (+4.0)	45.6 (+2.4)	35.9 (+7.3)	56.5	45.2
TÜLU 🐫 30B	57.7 (+3.1)	53.0 (+17.0)	51.9 (+2.4)	51.9 (-3.4)	48.0 (+5.2)	62.3	54.1
TÜLU 🐫 65B	59.2 (+0.5)	59.0 (+9.0)	54.4 (-3.7)	56.6 (-0.2)	49.4 (+2.5)	61.8	56.7
camel models trained on our final Human+GPT data mixture using LLAMA-2 ↓							
TÜLU-1.1 🐫 7B	49.2 (+7.4)	37.0 (+25.0)	44.2 (+4.9)	52.8 (+1.6)	33.9 (+7.1)	57.3	45.7
TÜLU-1.1 🐫 13B	52.3 (+0.3)	53.0 (+28.0)	50.6 (+1.7)	58.8 (+2.3)	38.9 (+7.4)	64.0	52.9

Limitations of Instruction-Tuning

- ✓ Human-written pairs are very expensive (↖)
- ✓ Mismatch between LM objectives and human preferences

- factual error vs. imprecise adjectives



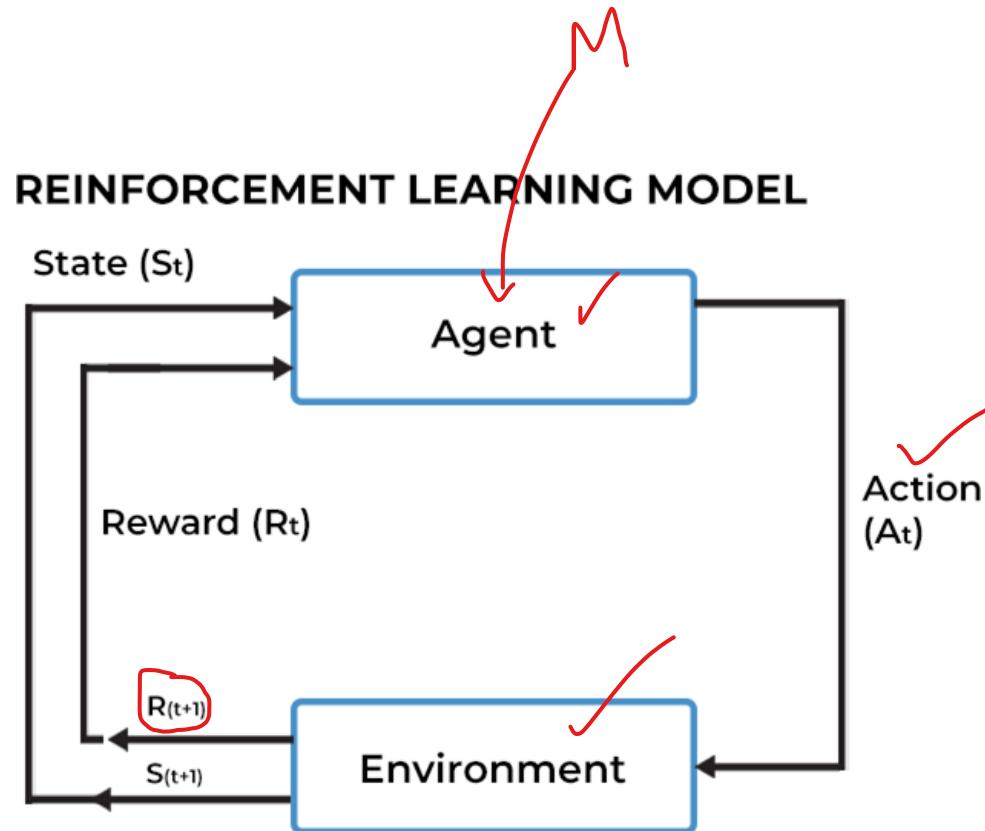
Common Objectives of Learning from Human Feedback

- ✓ • Align model output with our values ✓
- ✓ • Trustworthy and robust on factualness
- ✓ • Fairness on social values
- ✗ • Explainable with logical rationales

Content

- **InstructGPT (Proximal Policy Optimization)**
- Direct Preference Optimization
- Fine-Grained Human Feedback
- Open problems for RLHF

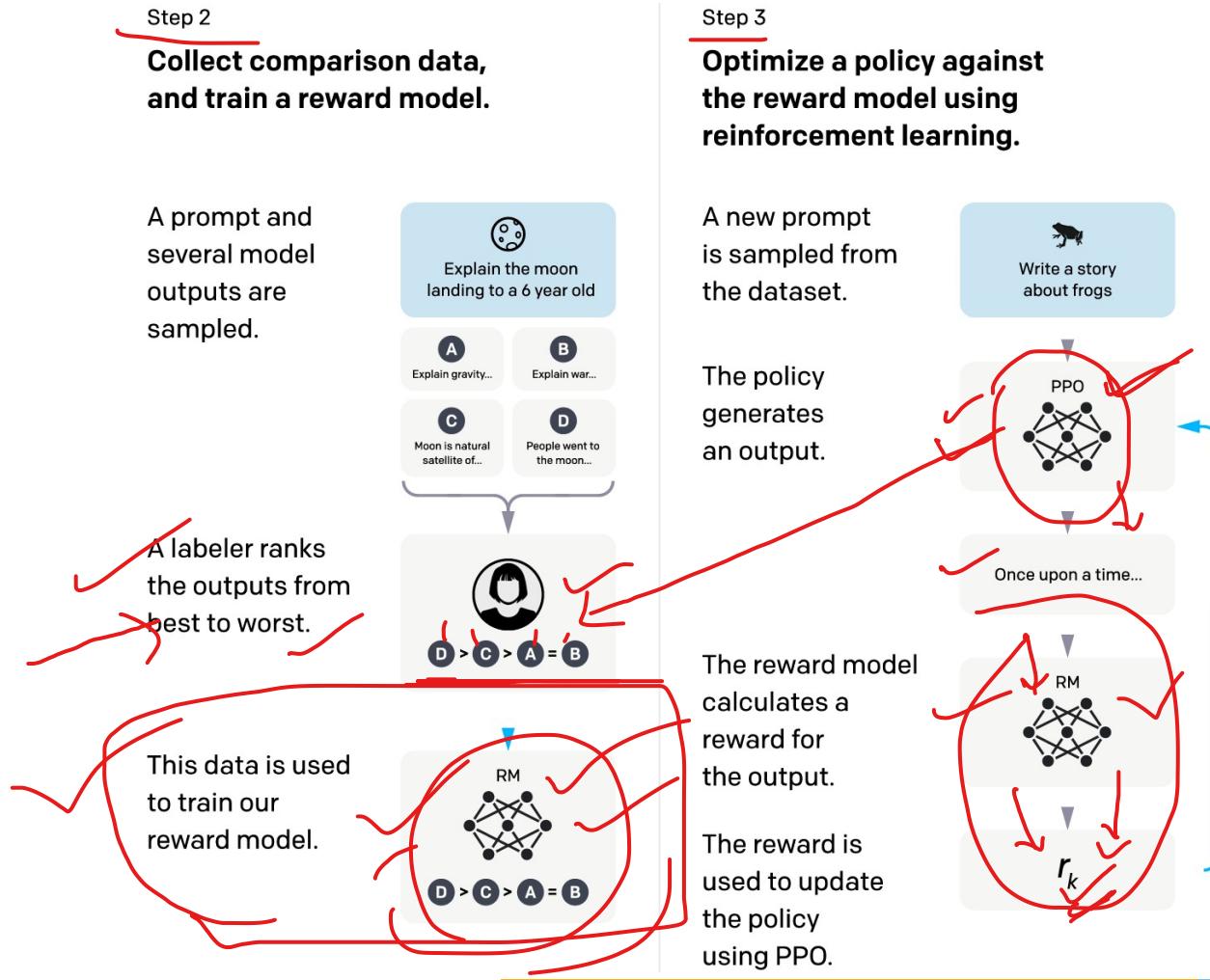
Reinforcement Learning Model



- An agent has a policy function, which can take action A_t according to the current state S_t .
- As a result of the action, the agent receives a reward R_t from the environment and transit to the next state S_{t+1} .

InstructGPT: Training language models to follow instructions with human feedback

(Ouyang et. al, 2022)

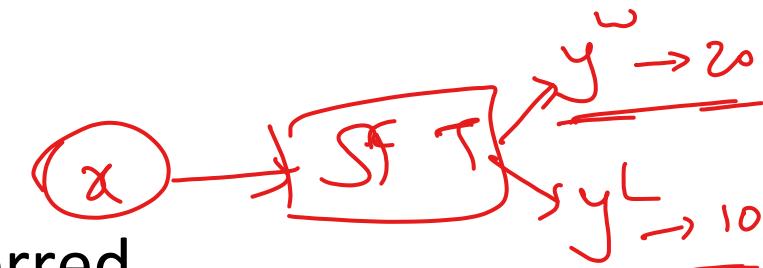


- Agent: language model
- Action: predict the next token
- Policy π_θ : the output distribution of the next token
- Reward: a reward model trained by human evaluations on model responses, so no more human-in-the-loop is needed r_ϕ

Reward Model Training

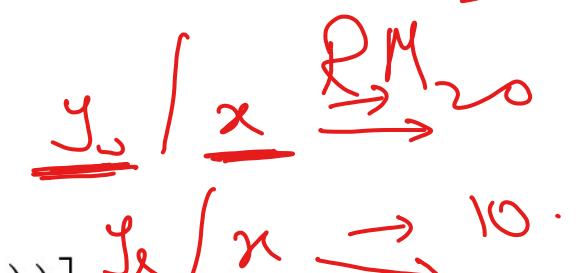
- Prompt supervised fine-tuned language model with to produce pairs of answers

$$(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$$



- Human annotators decide which one wins / is preferred

$$y_w \succ y_l \mid x$$



- A reward model is trained to score y_w higher than y_l

$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))]$

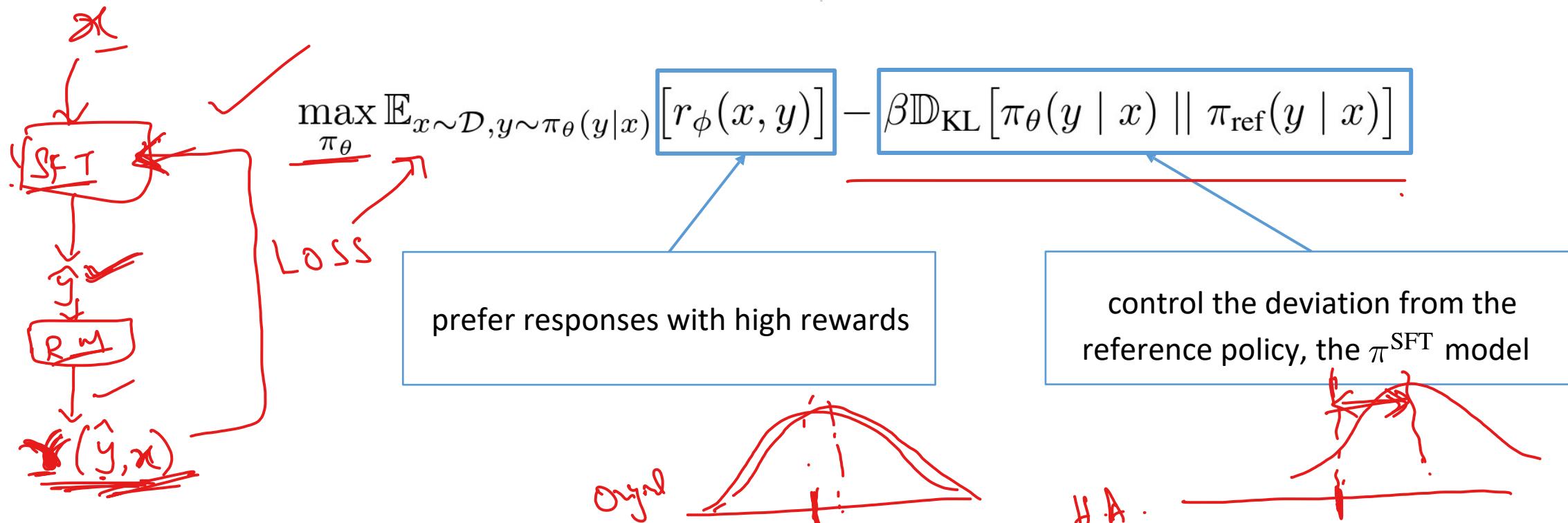
- A reward model is often initialized from π^{SFT} with a linear layer to produce a scalar reward value



~~I cooked a meal~~ → "I cooked a meal"

Fine-Tuning with RL: PPO

- Optimize the language model π_θ with feedback from the reward model r_ϕ



[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Fine-Tuning with RL: PPO

- Optimize the language model π_θ with feedback from the reward model r_ϕ

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi_\theta(y | x) || \pi_{\text{ref}}(y | x)]$$

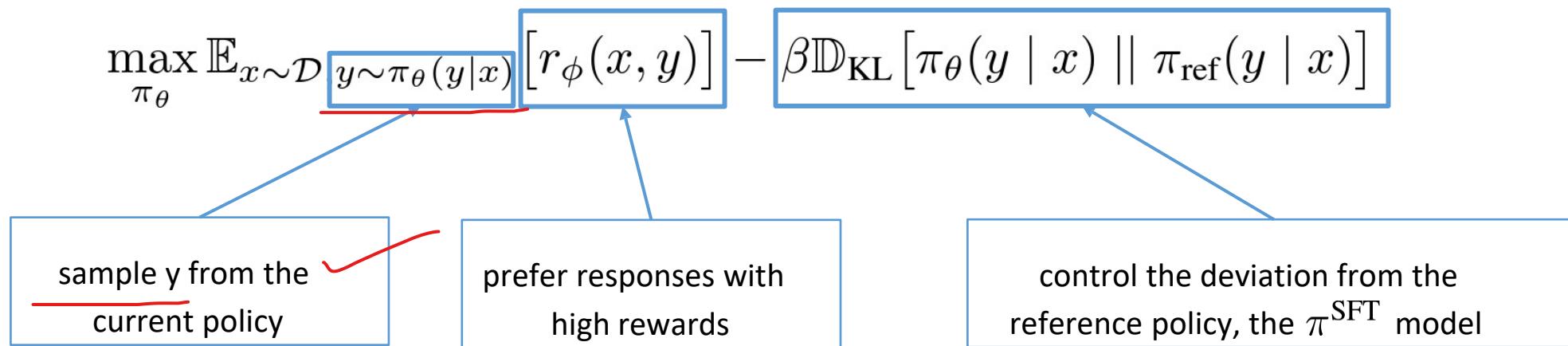
prefer responses with high rewards
control the deviation from the reference policy, the π^{SFT} model

- ✓ prevent mode-collapse to single high reward answers
- ✓ prevent the model deviating too far from the distribution where the reward model is accurate

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Fine-Tuning with RL: PPO

- Optimize the language model π_θ with feedback from the reward model r_ϕ



- prevent mode-collapse to single high reward answers
- prevent the model deviating too far from the distribution where the reward model is accurate

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Fine-Tuning with RL: PPO-px

- Training objective

$$E_{(x,y) \sim D_{\pi_\phi^{\text{RL}}}} [r_\theta(x, y) - \beta \log \left(\frac{\pi_\theta(y | x)}{\pi_{\text{ref}}(y | x)} \right)] + \gamma E_{x \sim D_{\text{pretrain}}} [\log(\pi_\theta(x))]$$

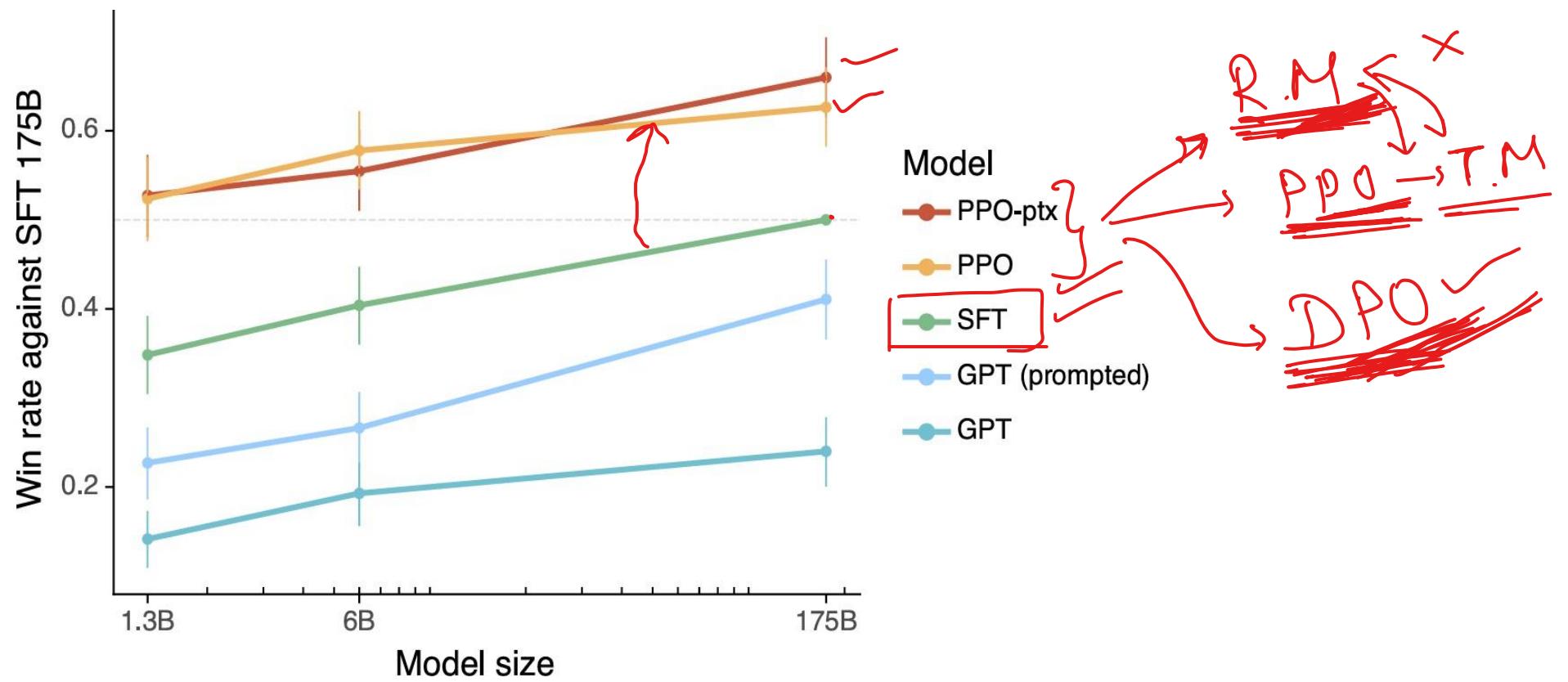
The equation is annotated with red markings: a checkmark above the first term, a circled '0' over the gamma term, and a checkmark next to the final term.

- Add pre-training gradients to fix the performance regressions on public NLP tasks
- For PPO models, γ is set to 0

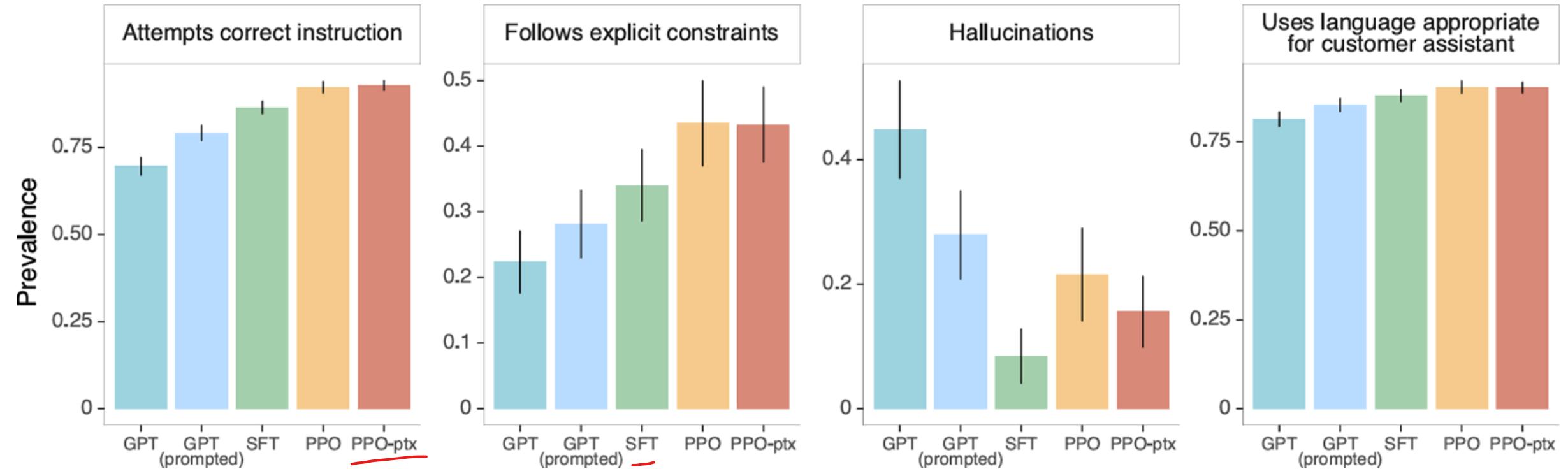
[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Comparison with Baselines

- RLHF models are more preferred by human labelers



Evaluations on Different Aspects



Limitation of PPO methods

- Need to train multiple models: a reward model and a policy model ✓ *Transformer / LM*
- Need sampling from LM during fine-tuning . *Computationally Expensive*
- The RL training process is too complicated! → 2
- Is it possible to directly train a language model from the human preference annotations?

DPO: Direct Preference Optimization: Your Language Model is Secretly a Reward Model

(Rafailov et. al, 2023)

- Looking into the PPO objective

$$\max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x)]$$

- Deriving optimal closed-form solution

$$\begin{aligned}
 & \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi} [r(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi(y|x) \parallel \pi_{\text{ref}}(y|x)] \\
 &= \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \\
 &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \\
 &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x) \right]
 \end{aligned}$$

partition function:
 $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

Direct Preference Optimization

- PPO Objective

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x) \right]$$

partition function:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- Partition function is a function of only x and depend on the policy $\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$, but does not
- Therefore we can define $\pi^*(y|x)$ valid probability, therefore the objective can be seen as a KL divergence between two probability distribution
- The optimal solution of the objective

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

Direct Preference Optimization

- Every reward function induce an optimal policy

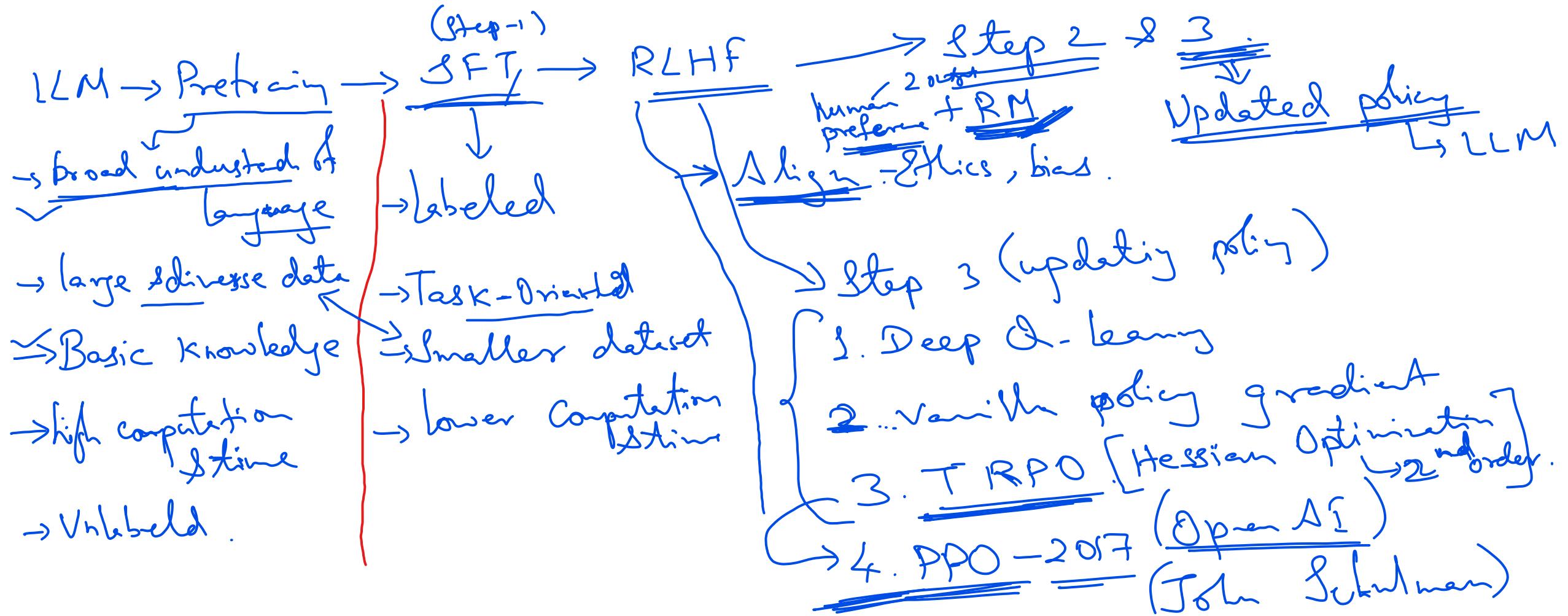
$$\pi_r(y | x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y | x) \exp \left(\frac{1}{\beta} \underline{r(x, y)} \right)$$

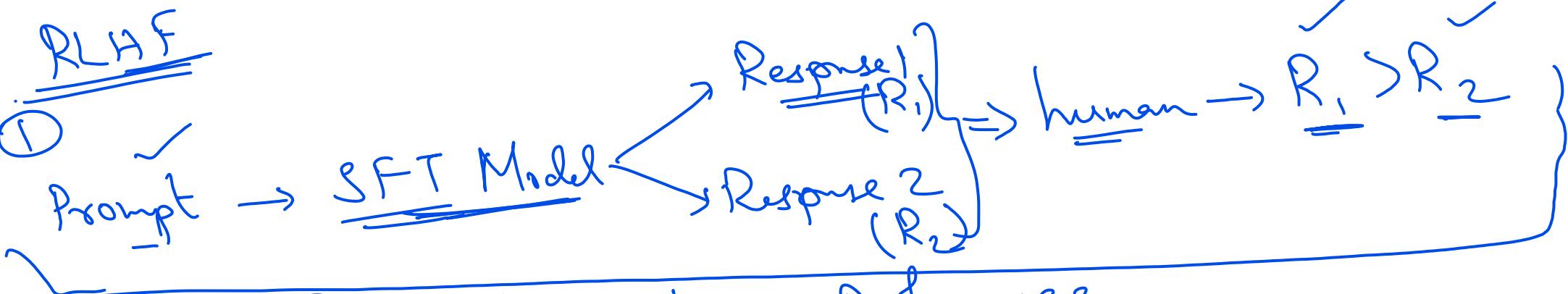
- Every policy is the optimal policy of some reward function

$$r(x, y) = \beta \log \frac{\pi_r(y | x)}{\pi_{\text{ref}}(y | x)} + \boxed{\beta \log Z(x)}$$

This term is intractable!

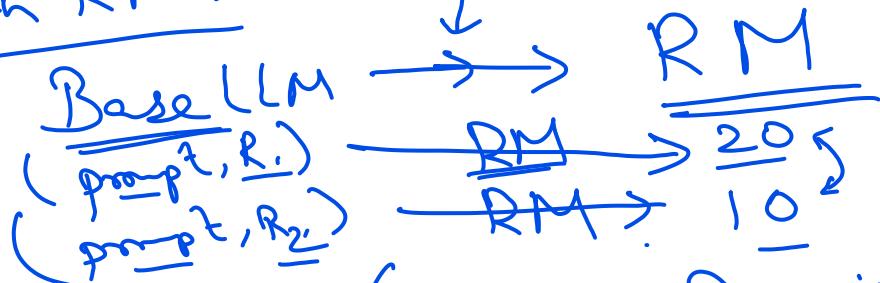
- Key idea: train the policy model so that $r(x, y)$ fits the human preference data!



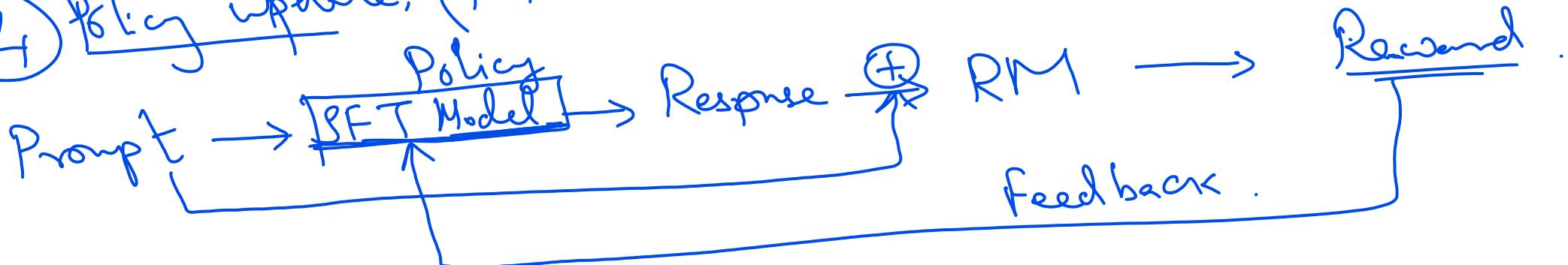


② Collect human preferences.

③ Train RM.



④ Policy update:- (PPO - Proximal Policy opt.)



Proximal Policy Optimization Algorithms

Points from the PPO paper (Proximal Policy Optimization Algorithms by OpenAI, 2017 - <https://arxiv.org/pdf/1707.06347>):

- Q-learning (with function approximation) fails on many simple problems and is poorly understood.
- Vanilla policy gradient methods have poor data efficiency and robustness.
- Trust region policy optimization (TRPO) is relatively complicated and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks).
- PPO adds a soft constraint that can be optimized by a first-order optimizer. We may make some bad decisions once a while, but it strikes a good balance on the speed of the optimization.

Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0 , initial KL penalty β_0 , target KL-divergence δ
for $k = 0, 1, 2, \dots$ **do**

Collect set of partial trajectories \mathcal{D}_k on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta || \theta_k)$$

by taking K steps of minibatch SGD (via Adam)

if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \geq 1.5\delta$ **then**

$$\beta_{k+1} = 2\beta_k$$

else if $\bar{D}_{KL}(\theta_{k+1} || \theta_k) \leq \delta/1.5$ **then**

$$\beta_{k+1} = \beta_k/2$$

end if

end for

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0 , clipping threshold ϵ

for $k = 0, 1, 2, \dots$ **do**

Collect set of partial trajectories \mathcal{D}_k on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Compute policy update

$$\theta_{k+1} = \arg \max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[\sum_{t=0}^T \left[\min(r_t(\theta) \hat{A}_t^{\pi_k}, \underbrace{\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t^{\pi_k}}_{\Sigma = 0.2}) \right] \right]$$

end for

$\Sigma = 0.2$

DPO: Direct Preference Optimization: Your Language Model is Secretly a Reward Model

(Rafailov et. al, 2023)

- Looking into the PPO objective

$$\checkmark \max_{\pi_\theta} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x)]$$

- Deriving optimal closed-form solution

$$\begin{aligned}
 & \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi} [r(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi(y|x) \parallel \pi_{\text{ref}}(y|x)] \\
 &= \max_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[r(x, y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right] \\
 &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x, y) \right] \\
 &= \min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x) \right]
 \end{aligned}$$

 partition function:
 $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

Direct Preference Optimization

- PPO Objective

$$\min_{\pi} \mathbb{E}_{x \sim \mathcal{D}} \mathbb{E}_{y \sim \pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)} - \log Z(x) \right]$$

partition function:

$$Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- Partition function is a function of only x π_{ref} , but does not and depend on the policy π
- Therefore, we can define $\pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

$\pi^*(y|x)$ is a valid probability, therefore the objective can be seen as a KL divergence between two probability distribution

- The optimal solution of the objective $\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

Direct Preference Optimization

- Every reward function induce an optimal policy

$$\checkmark \quad \underline{\pi_r(y | x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y | x) \exp\left(\frac{1}{\beta} r(x, y)\right)}$$

- Every policy is the optimal policy of some reward function

$$r(x, y) = \beta \log \frac{\pi_r(y | x)}{\pi_{\text{ref}}(y | x)} + \boxed{\beta \log Z(x)}$$

This term is intractable!

not dependent on model.

- Key idea: train the policy model so that $r(x, y)$ fits the human preference data!

Direct Preference Optimization

$\gamma_\theta + KL$

- Recall the reward model training loss

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} [\log \sigma(\underline{r_\phi(x, y_w)} - \underline{r_\phi(x, y_l)})]$$

R_1 R_2

- The partition function cancels out when we take the difference between the reward of a pair of responses!
- DPO training objective:

✓ $\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$

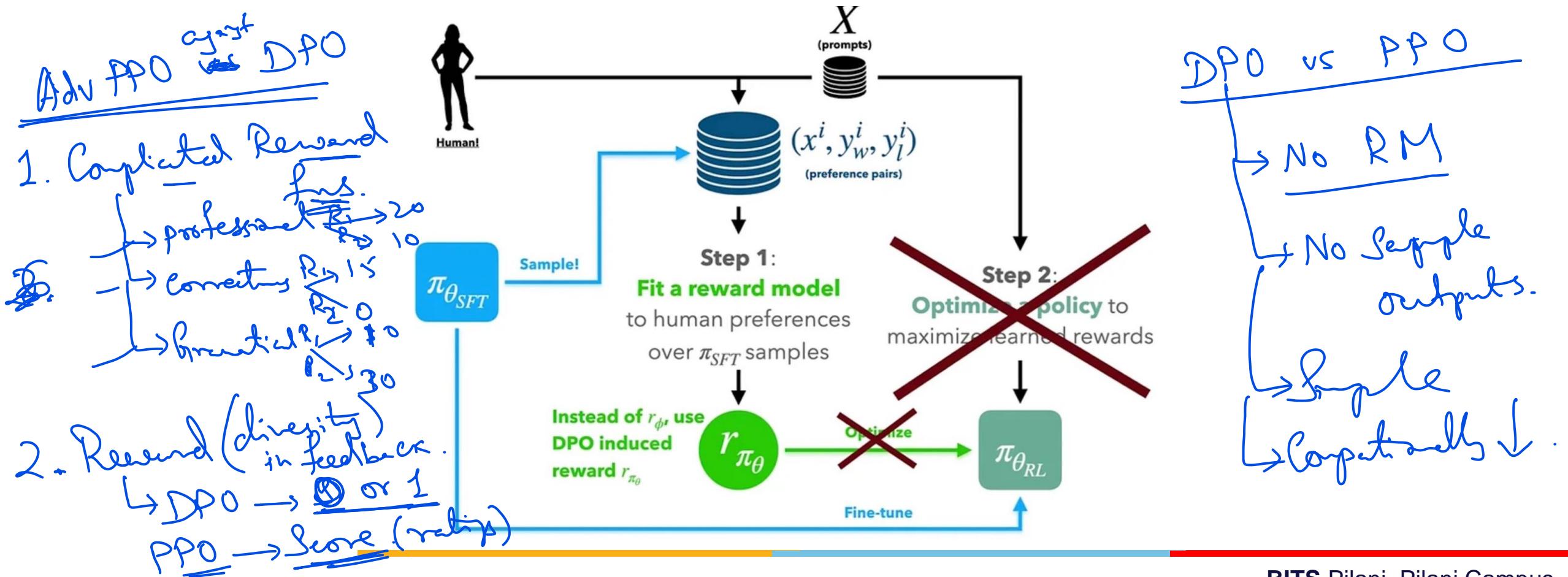
- A simple classification loss!

γM

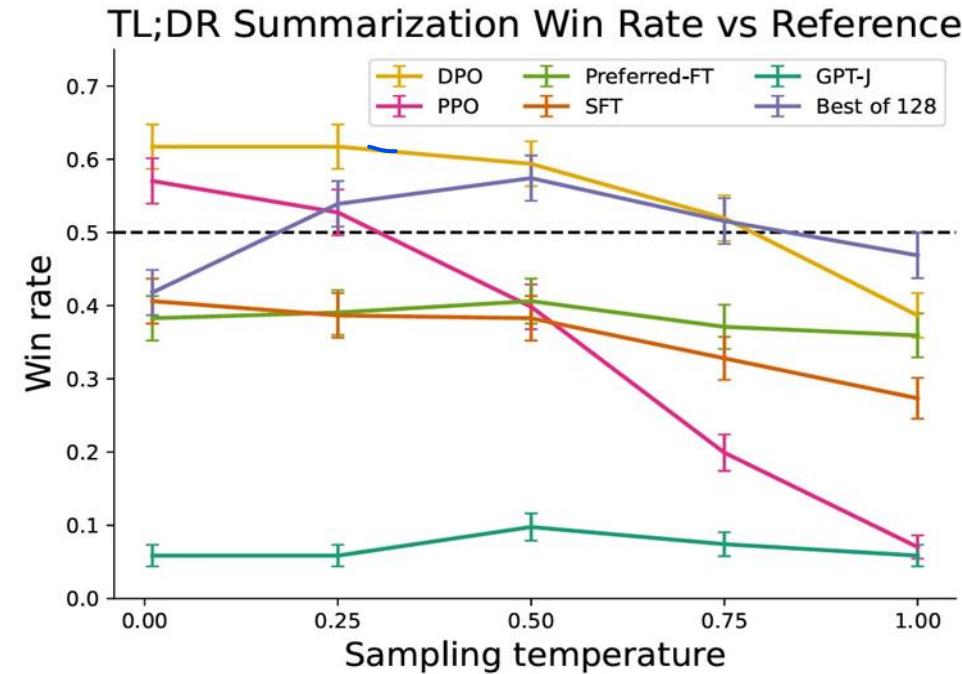
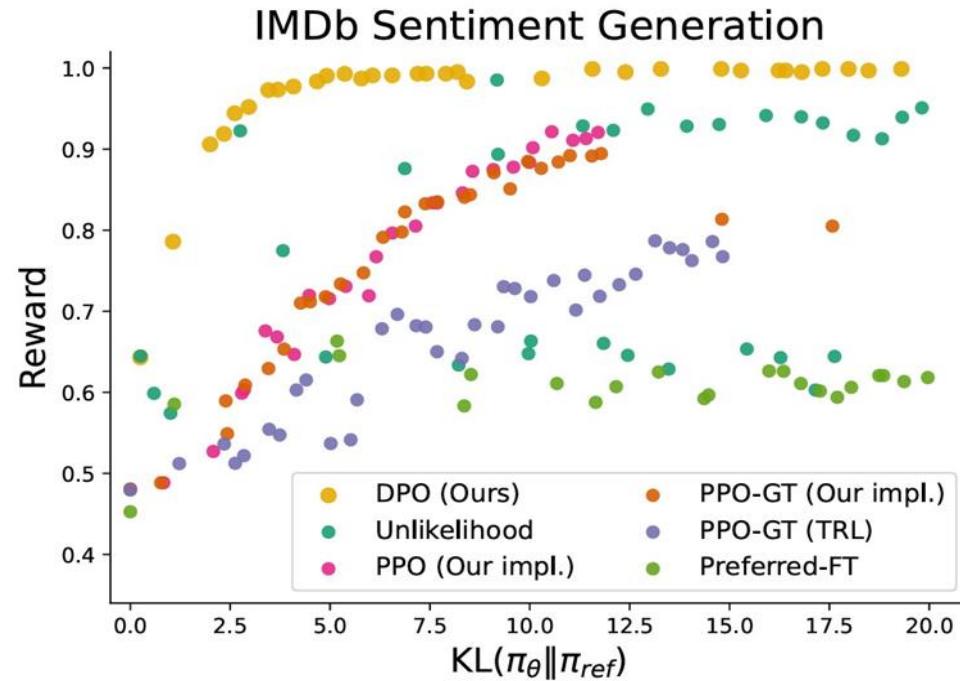
Cross Entropy

What does DPO do?

- DPO eliminates the need to train a reward model, sample from the LLM during fine-tuning, or perform significant hyperparameter search.



Comparison with Baseline Models



- Preferred-FT: Fine-tune the model on y_w
- PPO-GT: reward model is the ground truth of the sentiment
- Unlikelihood: optimize the policy model to maximize $P(y_w)$ and minimize $P(y_l)$
- Best of N: sampling N responses from the SFT model (very inefficient)

Comparison between PPO and DPO

- DPO training is cheaper and more stable than PPO training
 - PPO can handle more informative human feedback (e.g., numerical ratings) while DPO can only handle binary signals
-

Fine-Grained Human Feedback Gives Better Rewards for Language Model Training

(Wu et. al, 2023)

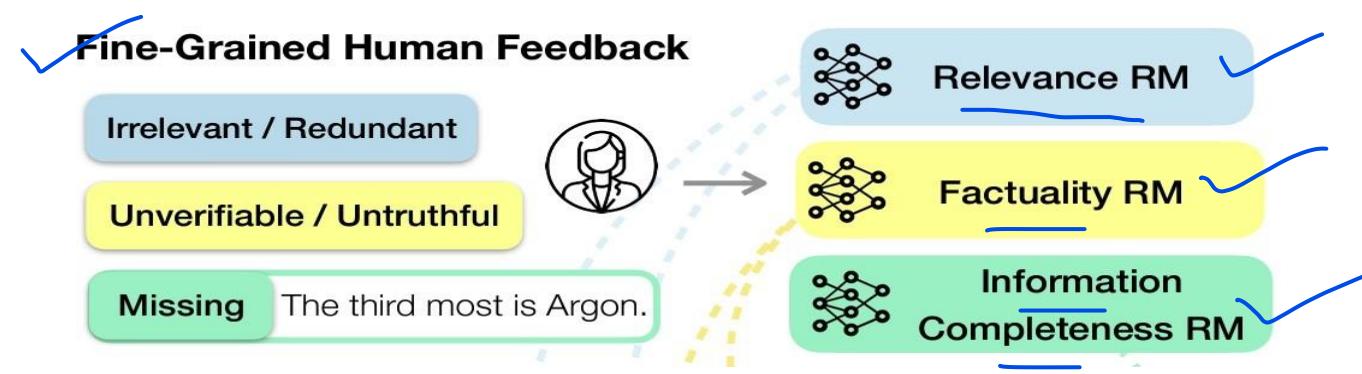
- Assigning a single score to the model output may not be informative enough

Prompt:

What are the 3 most common gasses in earth's atmosphere?

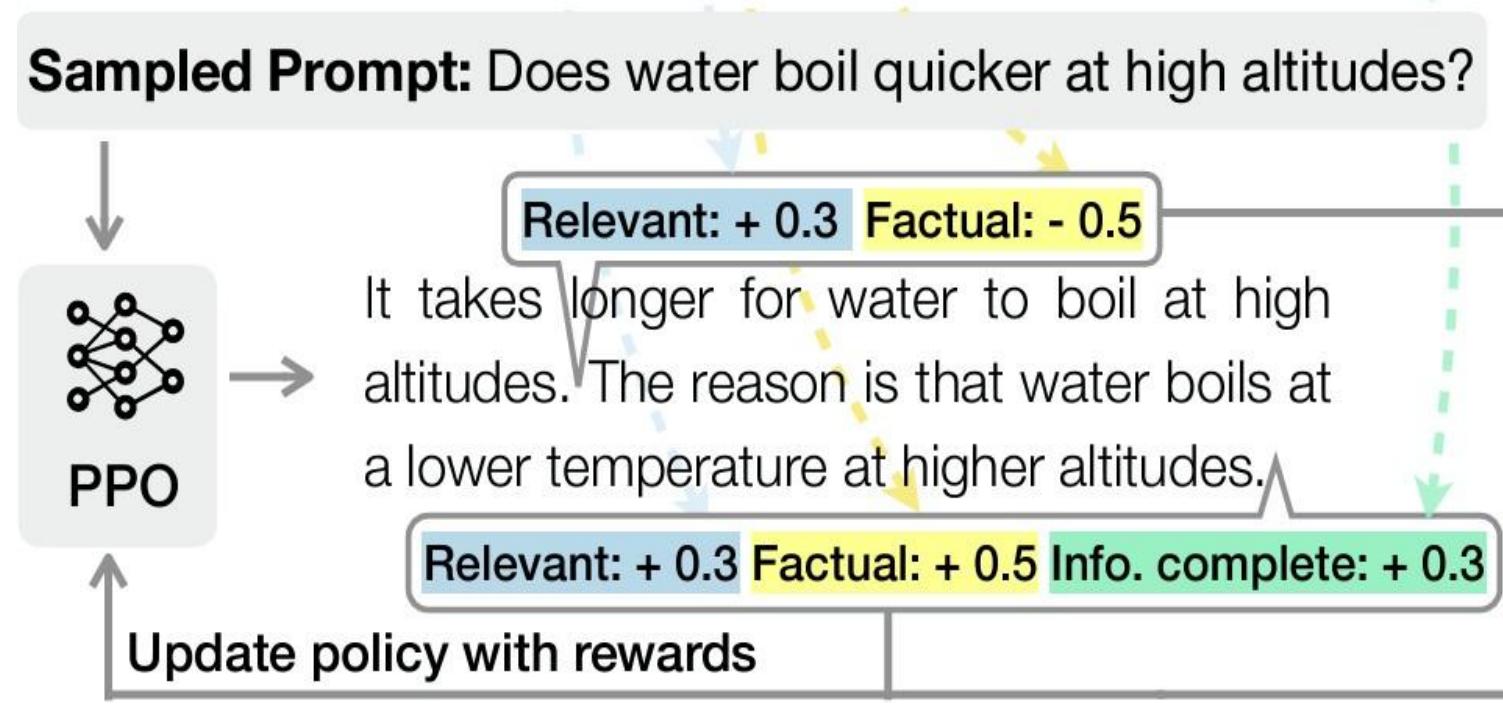
LM output:

The atmosphere of Earth is a layer of gases retained by Earth's gravity. The most common gas, by dry air volume, is nitrogen. The second most is oxygen. The third most is carbon dioxide.



Multiple Fine-Grained Reward Functions

- (1) Provide a reward after every segment (e.g., a sentence) is generated
- (2) Different feedback types: factual incorrectness, irrelevance, and information incompleteness



Combined Reward Function

$$\underline{r}_t = \sum_{k=1}^K \sum_{j=1}^{L_k} \left(\mathbb{1}(t = T_j^k) \underbrace{w_k R_{\phi_k}(x, y, j)}_{\text{---}} \right) - \beta \log \frac{P_{\theta}(a_t | s_t)}{P_{\theta_{\text{init}}}(a_t | s_t)}$$

- w_k is a weight assigned to each reward function

Use Case I: Detoxification

- Perspective API: measures toxicity (0: non-toxic, 1: toxic)

✓ **(a) Holistic Rewards for (non-)Toxicity**

$$\text{Reward} = 1 - 0.60 = 0.40$$

I am such an idiot. She is so smart!

Toxicity = 0.60

✓ **(b) Sentence-level (Fine-Grained) Reward for (non-)Toxicity**

$$\text{Sent1 reward} = 0.00 - 0.72 = -0.72$$

$$\text{Sent2 reward} = 0.72 - 0.60 = 0.12$$

I am such an idiot. She is so smart!

Toxicity = 0.72

Toxicity = 0.60

Use Case I: Detoxification

- Learning from **denser** fine-grained reward is more sample efficient than holistic reward.
- Fine-grained reward locates where the toxic content is, which is a stronger training signal compared with a scalar reward for the whole text.

	Toxicity avg max (↓)	Fluency PPL (↓)	Diversity	
			dist-2 (↑)	dist-3 (↑)
GPT-2	0.192	9.58	0.947	0.931
Controlled Generation				
GeDi	0.154	24.78	0.938	0.938
DEXPERTS	0.136	22.83	0.932	0.922
Hol. RLHF	0.130	11.75	0.943	0.926
F.G. RLHF	0.081	9.77	0.949	0.932

Table 1: Results on the REALTOXICITYPROMPTS test set.

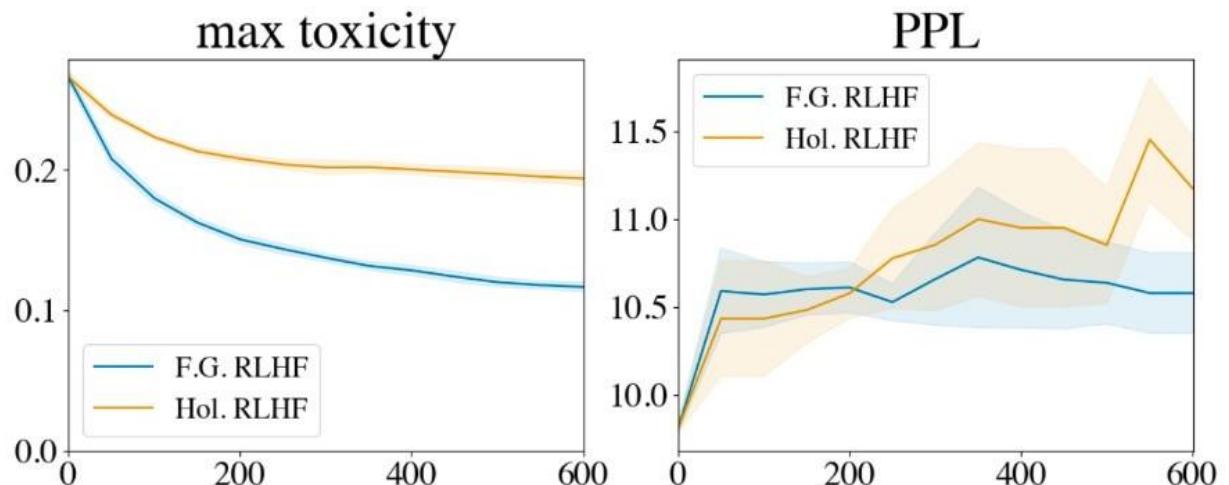


Figure 2: Curves of toxicity and perplexity on the dev set v.s. training steps.

Use Case II: Long-Form Question Answering

- Train a fine-grained reward model for each of the three aspects.

Prompt:

What are the 3 most common gasses in earth's atmosphere?

LM output:

The atmosphere of Earth is a layer of gases retained by Earth's gravity. The most common gas, by dry air volume, is nitrogen. The second most is oxygen. The third most is carbon dioxide.

Fine-Grained Human Feedback

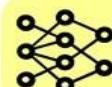
Irrelevant / Redundant

Unverifiable / Untruthful

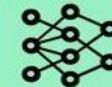
Missing The third most is Argon.



Relevance RM



Factuality RM



Information Completeness RM

Use Case II: Long-Form Question Answering

- Fine-Grained RLHF outperforms SFT and Preference RLHF on all error types.
- RLHF (both preference-based and fine-grained) are particularly effective in reducing factual errors.

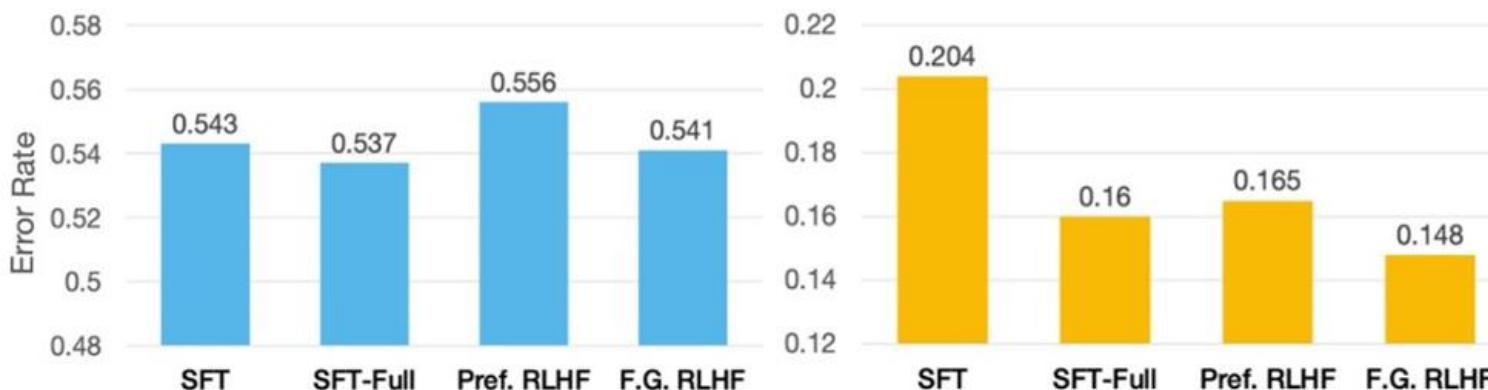


Figure 3: Human evaluation on *rel.* (left) and *fact.* (right) error, measured by % of sub-sentences that contain the error type (↓).

Ours v.s.	Win	Tie	Lose
SFT	23.0%	65.5%	11.5%
SFT-Full	22.0%	61.0%	17.0%
Pref. RLHF	19.5%	71.0%	9.5%

Table 2: Human pairwise comparison on information completeness (*comp.*), where win/lose refers to FINE-GRAINED RLHF.

Fine-Grained RLHF for Customizing LLM Behavior

- Relevance reward weight: 0.4/0.3/0.2
- Keep factualness/completeness reward weight fixed
- Relevance reward penalize referencing passages and auxiliary information

Question: | When did the French join revolution on colonists' side?

LM outputs with different reward model configurations:

Short: French involvement in the American Revolutionary War began in 1775, when France, a rival of the British Empire, secretly shipped supplies to the Continental Army. A Treaty of Alliance in 1778 soon followed, which led to shipments of money and matériel to the United States. Subsequently, the Spanish Empire and the Dutch Republic also began to send assistance, leaving the British Empire with no allies.

Medium: French involvement in the American Revolutionary War began in 1775, when France, a rival of the British Empire, secretly shipped supplies to the Continental Army. A Treaty of Alliance in 1778 soon followed, which led to shipments of money and matériel to the United States. Subsequently, the Spanish Empire and the Dutch Republic also began to send assistance, leaving the British Empire with no allies. On March 17, 1778, four days after a French ambassador informed the British government that France had officially recognized the United States as an independent nation, Great Britain declared war on France, thereby engaging that nation in the American Revolutionary War.

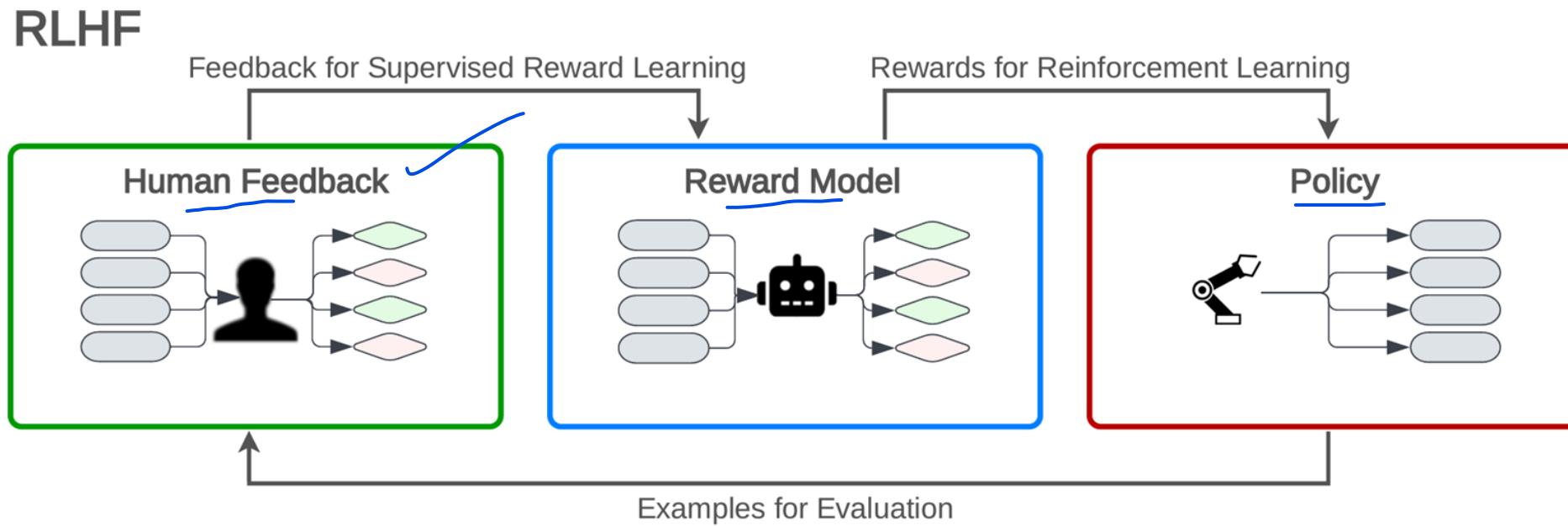
Long: French involvement in the American Revolutionary War began in 1775, when France, a rival of the British Empire, secretly shipped supplies to the Continental Army. A Treaty of Alliance in 1778 soon followed, which led to shipments of money and matériel to the United States. Subsequently, the Spanish Empire and the Dutch Republic also began to send assistance, leaving the British Empire with no allies. On March 17, 1778, four days after a French ambassador informed the British government that France had officially recognized the United States as an independent nation with the signing of the Treaty of Amity and Commerce and the Treaty of Alliance, Great Britain declared war on France, thereby engaging that nation in the American Revolutionary War.

Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback

(Casper et. al, 2023)



- Challenges within each step: human feedback, reward model and policy



Challenges with Obtaining Human Feedback

- Human evaluators may have biases ✓
 - Studies found that ChatGPT models became politically biased post RLHF.
- Good oversight is difficult
 - Evaluators are paid per example and may make mistakes given time constraints; poor feedback on evaluating difficult tasks
- Data quality
 - cost / quality tradeoff
- Tradeoff between richness and efficiency of feedback types
 - comparison-based feedback, scalar feedback, correction feedback, language feedback, ...

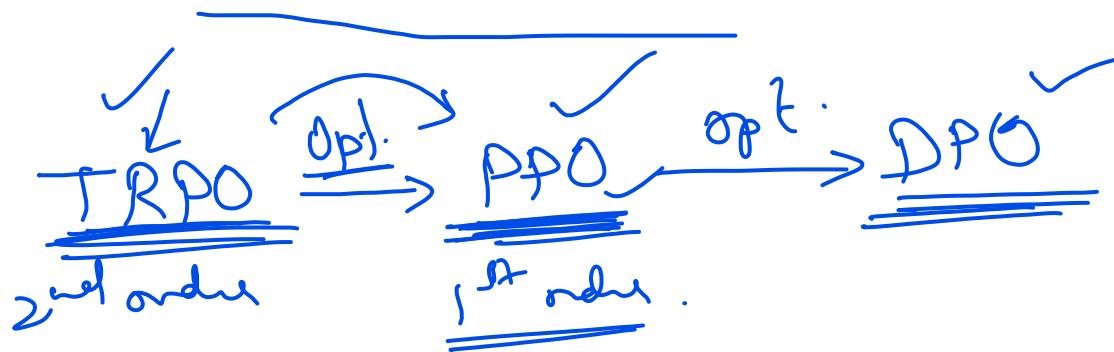
Challenges with the Reward Model

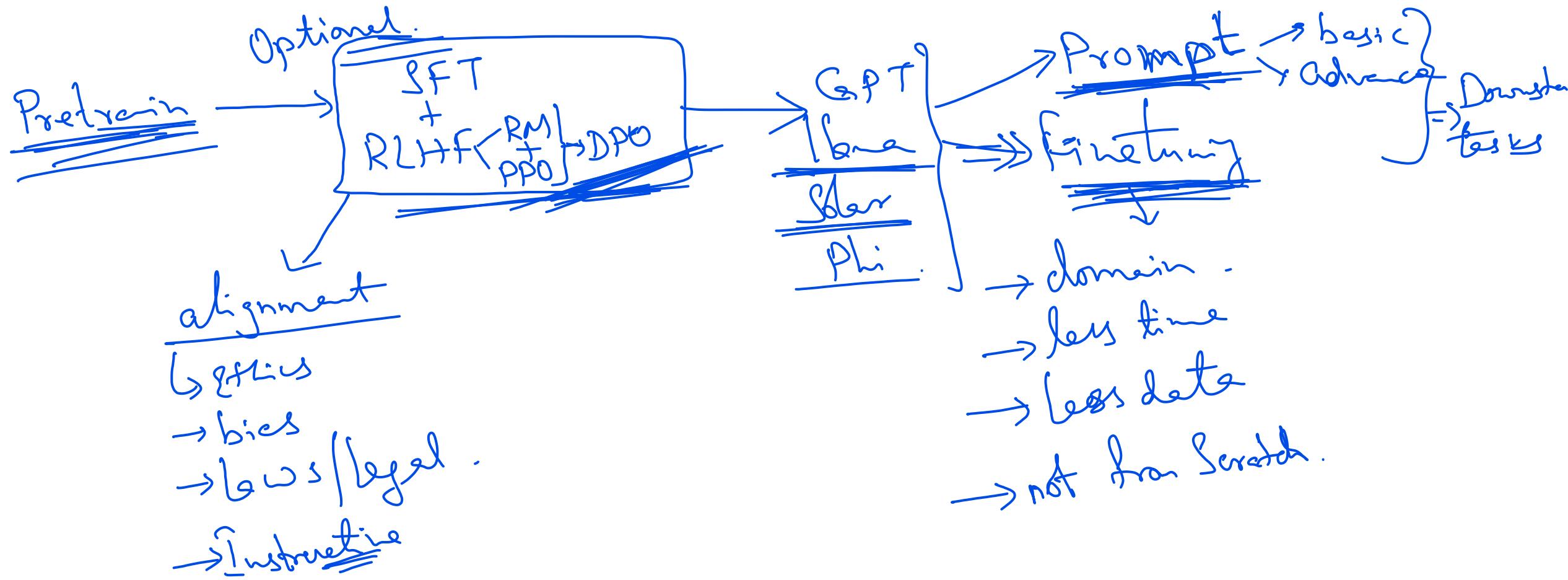
- A single reward function cannot represent a diverse society of humans
- Reward misgeneralization: reward models may fit with human preference data with unexpected features
- Evaluation of a reward model is difficult and expensive

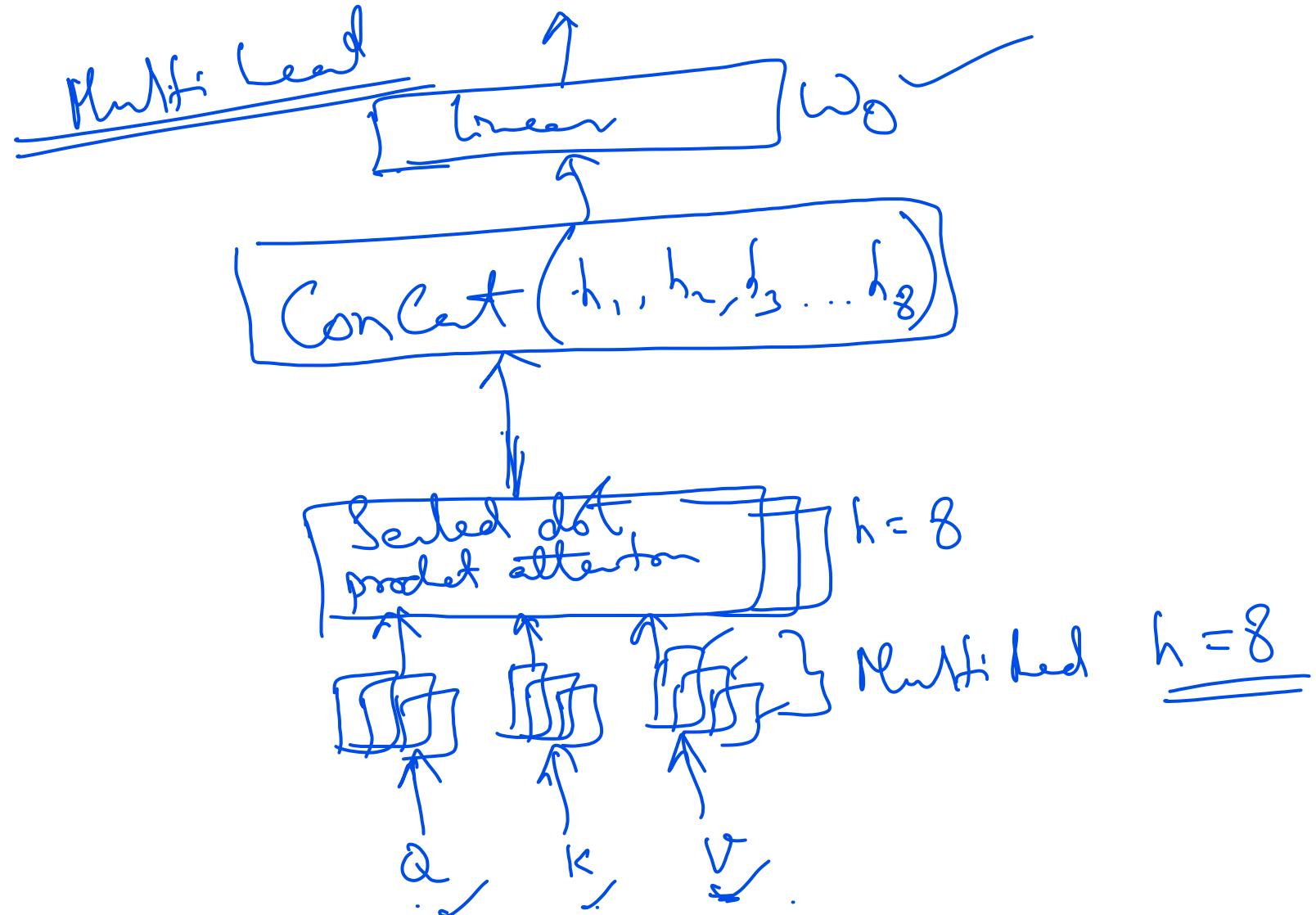
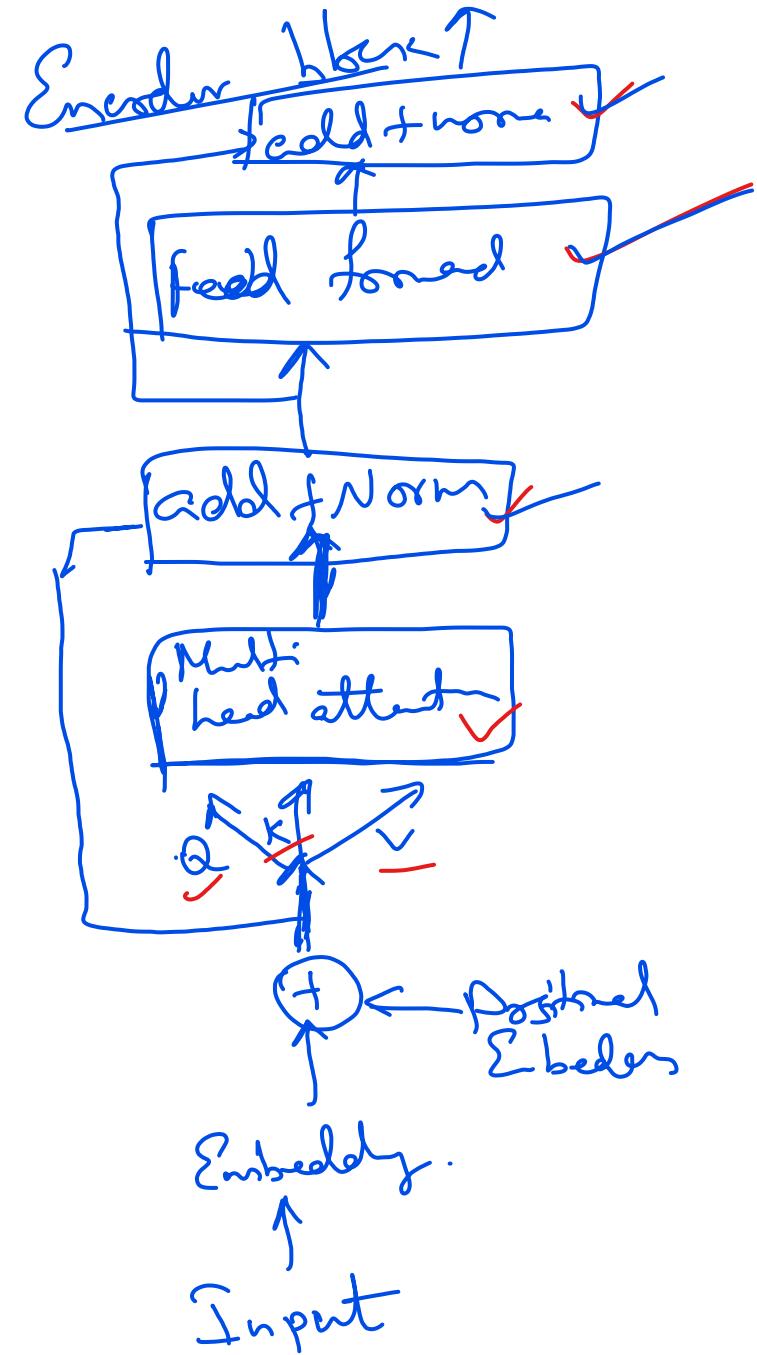
→ Costly → Expense (CPU, Storage)
→ Time

Challenges with the Policy

- Robust reinforcement learning is difficult
 - balance between exploring new actions and exploiting known rewards
 - the challenge intensifies in high-dimensional or sparse reward settings
- Policy misgeneralization: training and deployment environment is difference

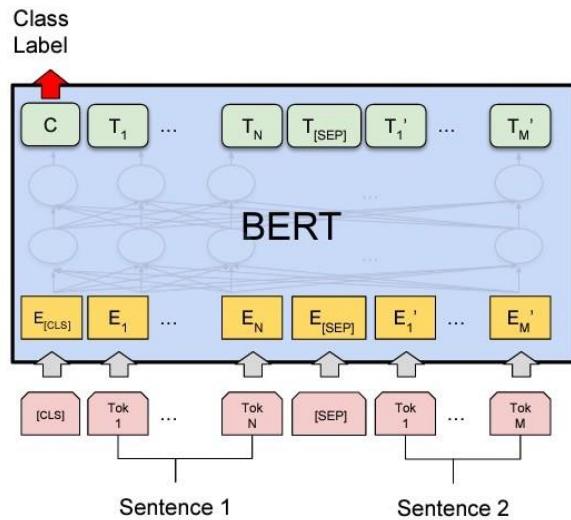




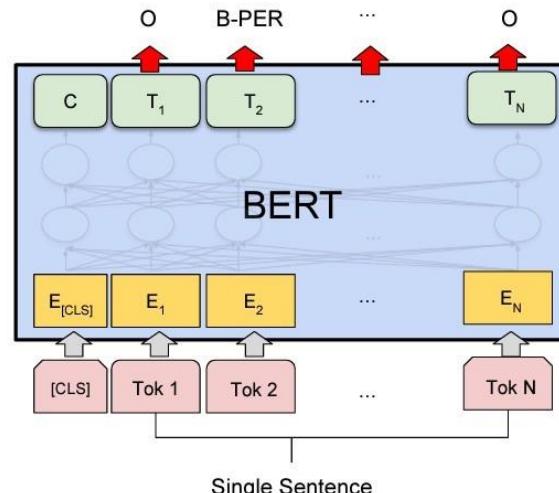


Background: Vanilla Fine-tuning

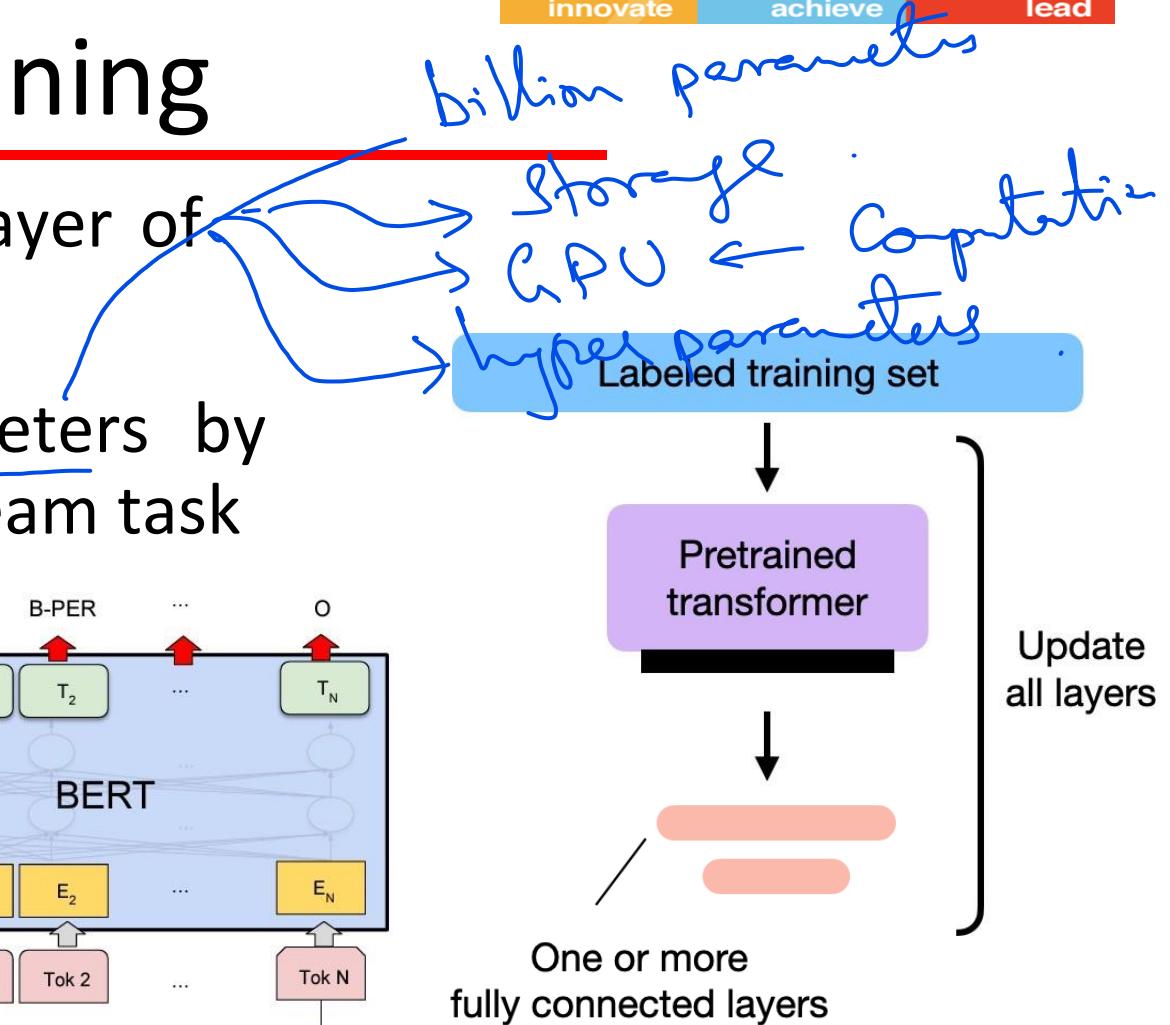
- Attach a task-specific layer to the last layer of the pre-trained transformer output
- Update the weights of all the parameters by backpropagating gradients on a downstream task



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

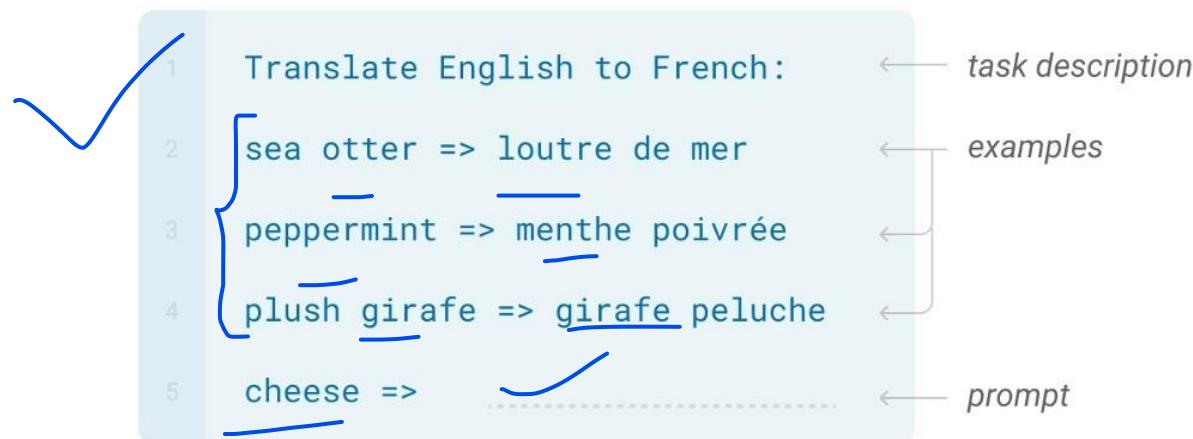
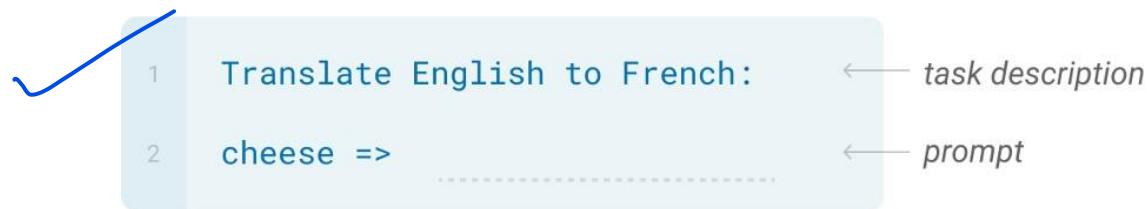


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER



Background: Prompting

- Prompting a language model with a natural description of the task, and possibly several few-shot examples. No gradient updates.



Vanilla Fine-Tuning vs. Prompting

- Vanilla fine-tuning
 - Pros: Can utilize more training data
 - Pros: Lead to stronger performance with more training data
 - Cons: Computationally expensive to train the complete network
 - Cons: Need to store a full set of model weights per task
- Prompting
 - Pros: Training-data efficient
 - Pros: Computational efficient
 - Cons: Performance depends on prompts and examples
 - Cons: Finding a good prompt could be challenging

Parameter-Efficient Fine-Tuning

- Rather than fine-tuning the parameters in the entire model, only fine-tune a small set of weights.
 - Addition: add a small external network for each task
 - Prompt-based Methods
 - Adapter-based Methods
 - Reparameterization: reparametrize the model parameter to be more efficient for training
 - ✓ • LoRA

Prompt Engineering

- Paraphrasing the task instruction
- Adding detailed examples
- For each new task, search over the possible sequence space to find the prompt with the best output performance. -> **computationally expensive!**
- Can we use a set of parameters to replace these prompts and train them with labeled sets?

~~Prompt Template:~~

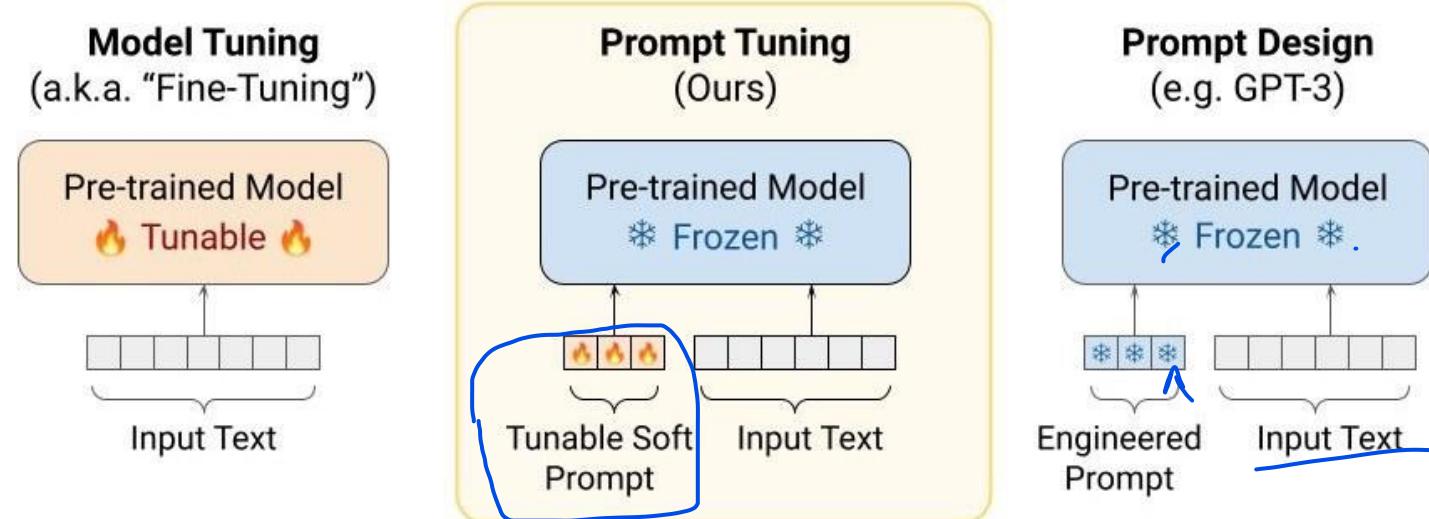
1. Identity
2. Style tokens
3. Handling Errors/outliers
4. Input [Dynamic]
5. Output format

→ JSON
→ Text
→ one word



Prompt-Tuning

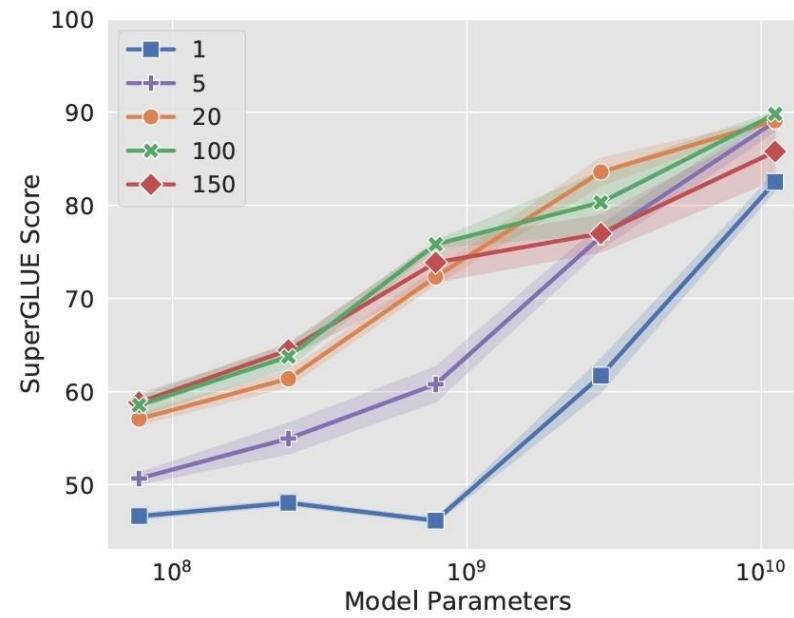
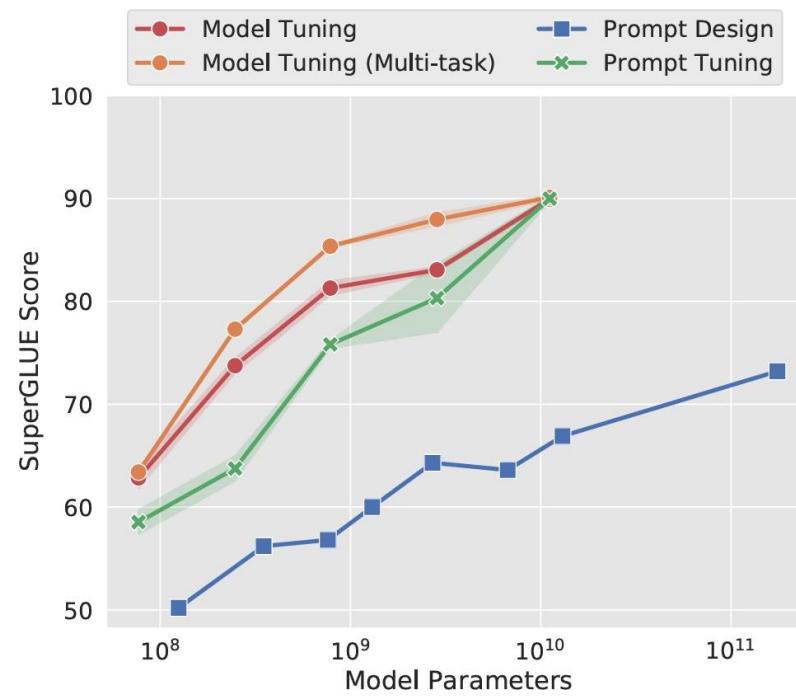
- Prepend a sequence of tokens as tunable embeddings to the input data (as soft prompts)
- freeze the whole Transformer model during training, and only tune the prepended soft prompts
- Only a small set of parameters need to be stored for each task



The Power of Scale for Parameter-Efficient Prompt Tuning

(Lester et. al, 2021)

- Prompt-tuning becomes more effective when the pre-trained model becomes larger
- Larger models perform well even with a small number of prompt tokens

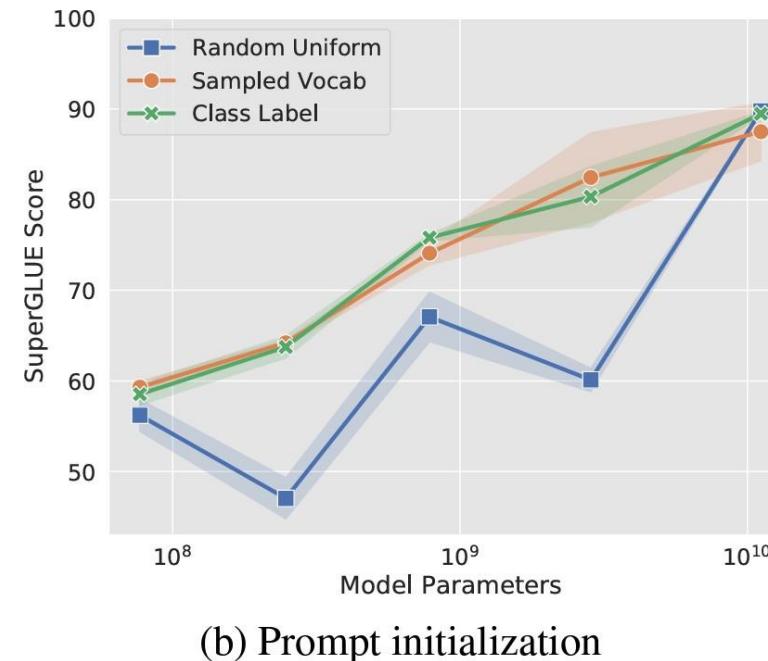


(a) Prompt length

The Power of Scale for Parameter-Efficient Prompt Tuning

(Lester et. al, 2021)

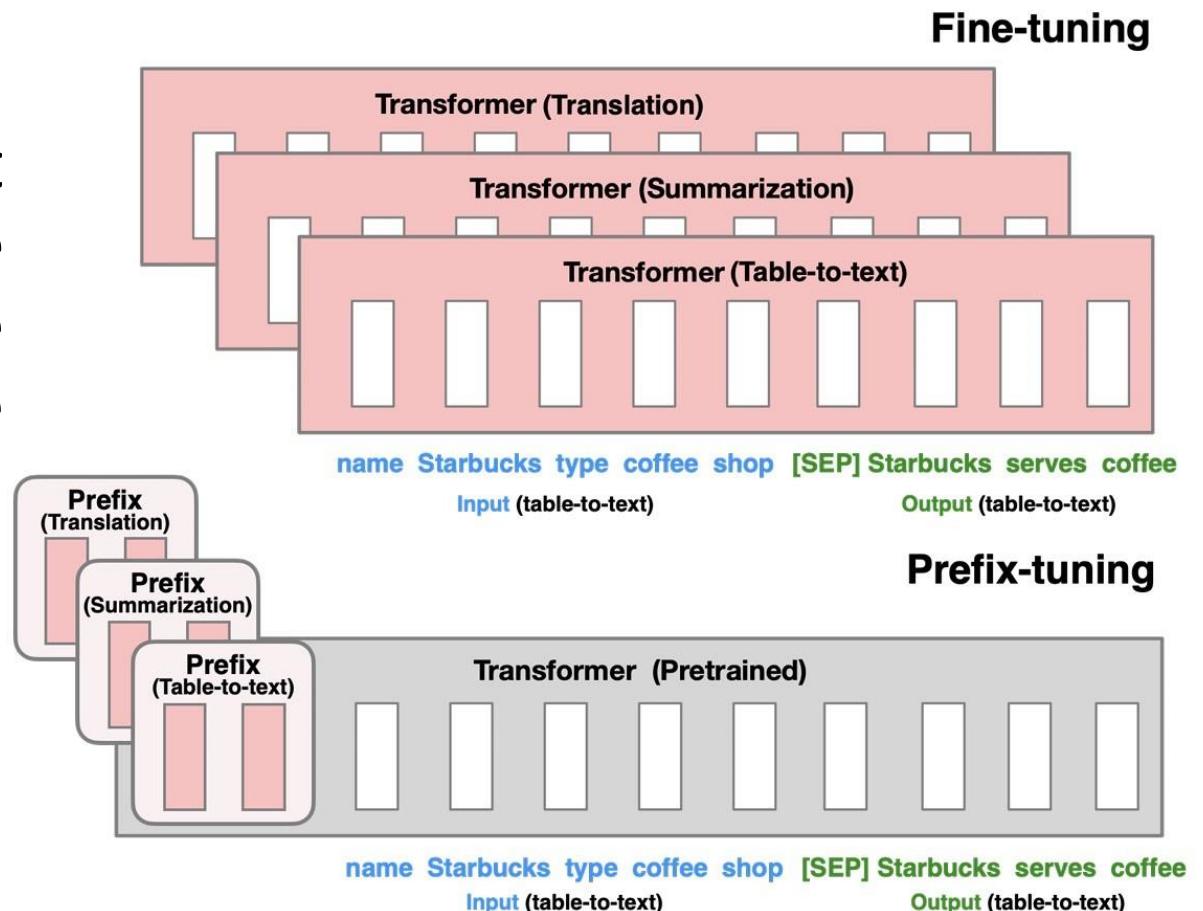
- Initializing prompt tokens with real tokens in vocabulary is helpful



Prefix-Tuning: Optimizing Continuous Prompts for Generation

(Li et. al, 2021)

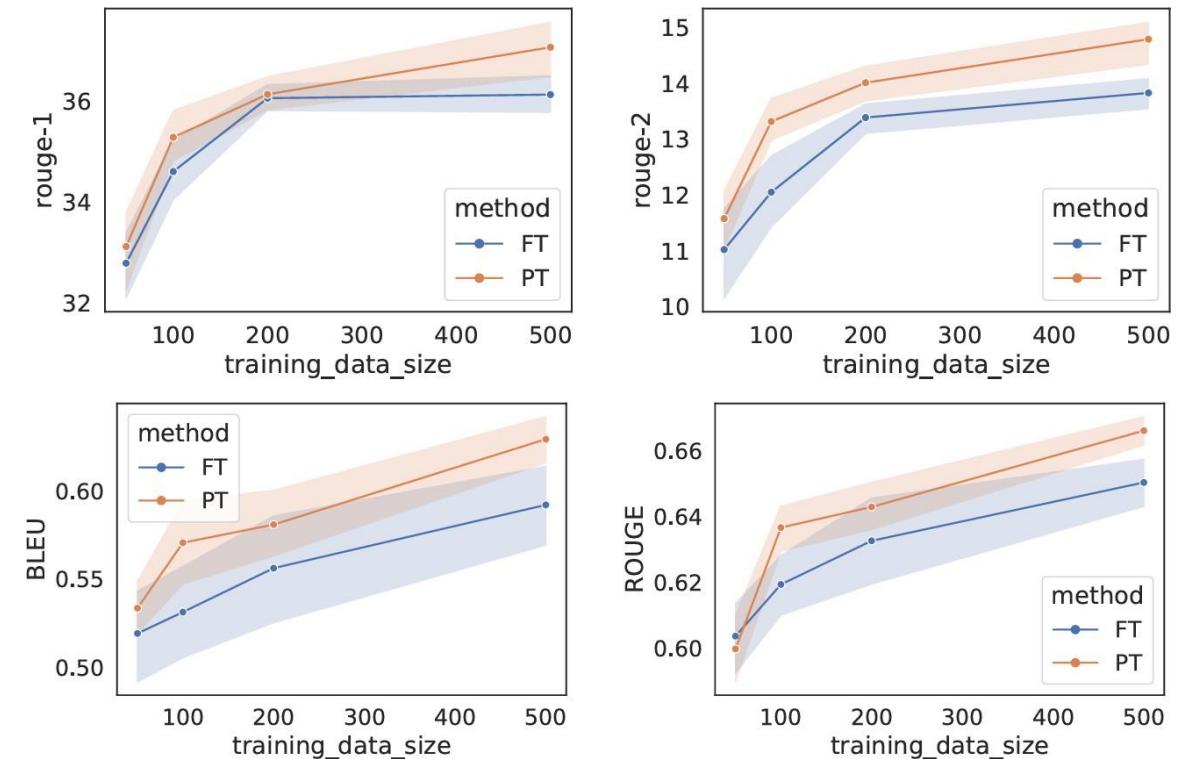
- Similar with prompt-tuning, except that the soft prompt tokens are prepended to each layer in the Transformer instead of just the input layer.



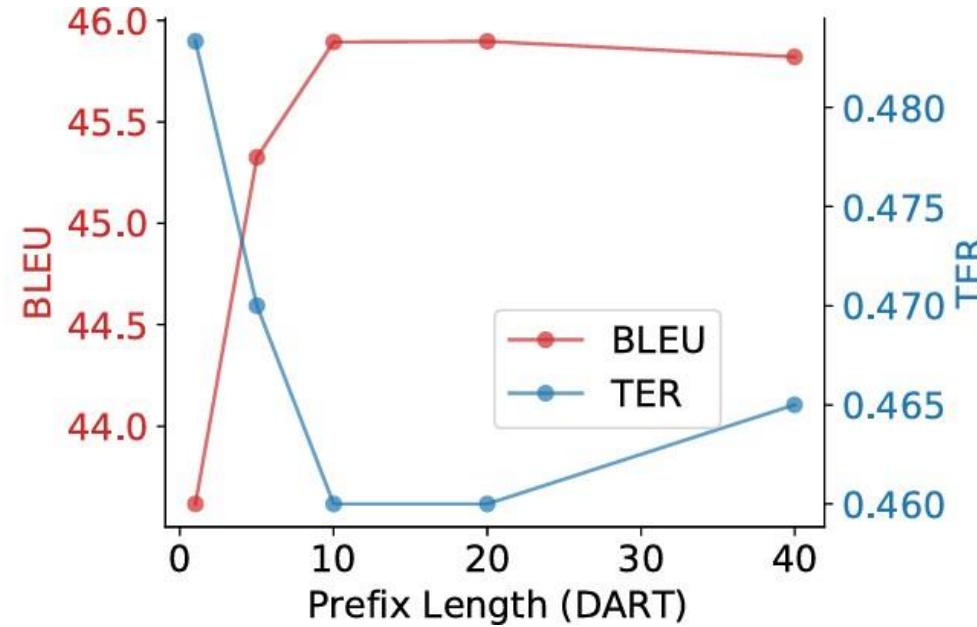
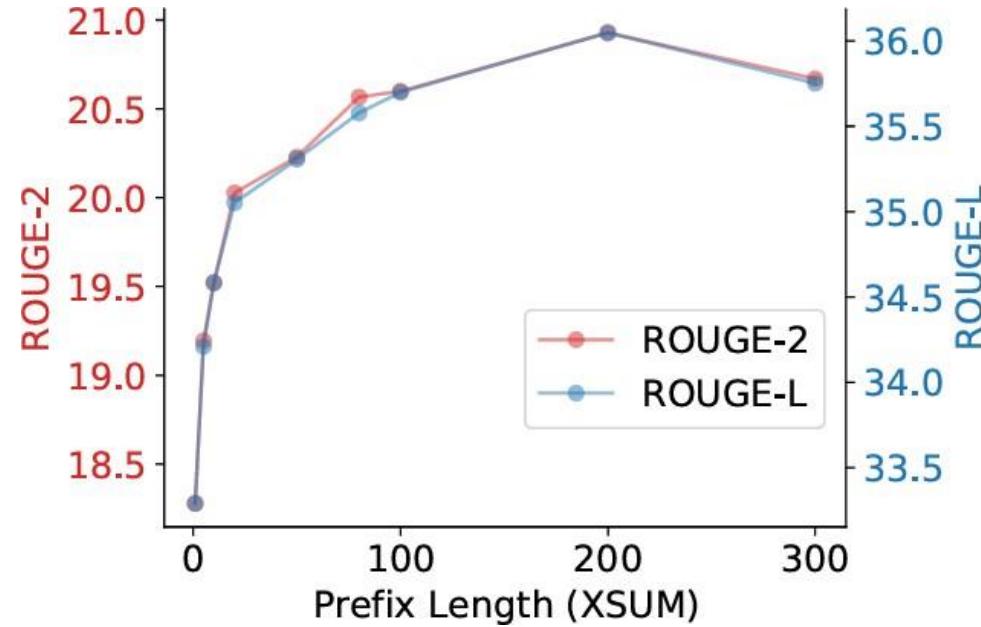
Experiments on Text Generation

- Prefix-tuning works better than fine-tuning under low-data settings.

Source	name : The Eagle type : coffee shop food : Chinese price : cheap customer rating : average area : riverside family friendly : no near : Burger King
Prefix (50)	The Eagle is a cheap Chinese coffee shop located near Burger King.
Prefix (100)	The Eagle is a cheap coffee shop located in the riverside near Burger King. It has average customer ratings.
Prefix (200)	The Eagle is a cheap Chinese coffee shop located in the riverside area near Burger King. It has average customer ratings.
Prefix (500)	The Eagle is a coffee shop that serves Chinese food. It is located in the riverside area near Burger King. It has an average customer rating and is not family friendly.
FT (50)	The Eagle coffee shop is located in the riverside area near Burger King.
FT (100)	The Eagle is a cheap coffee shop near Burger King in the riverside area. It has a low customer rating and is not family friendly.
FT (200)	The Eagle is a cheap Chinese coffee shop with a low customer rating. It is located near Burger King in the riverside area.
FT (500)	The Eagle is a cheap Chinese coffee shop with average customer ratings. It is located in the riverside area near Burger King.



Prefix Length



- Performance increases as the prefix length increases up to a threshold (200 for summarization and 10 for table-to-text) and then a slight performance drop occurs.

Prefix Initialization

- Random initialization leads to low performance with high variance.
- Initializing the prefix with real words significantly improves generation, as shown in Figure 5.
- Initializing with task relevant words such as “summarization” and “table-to-text” obtains slightly better performance than task irrelevant words such as “elephant” and “divide”.

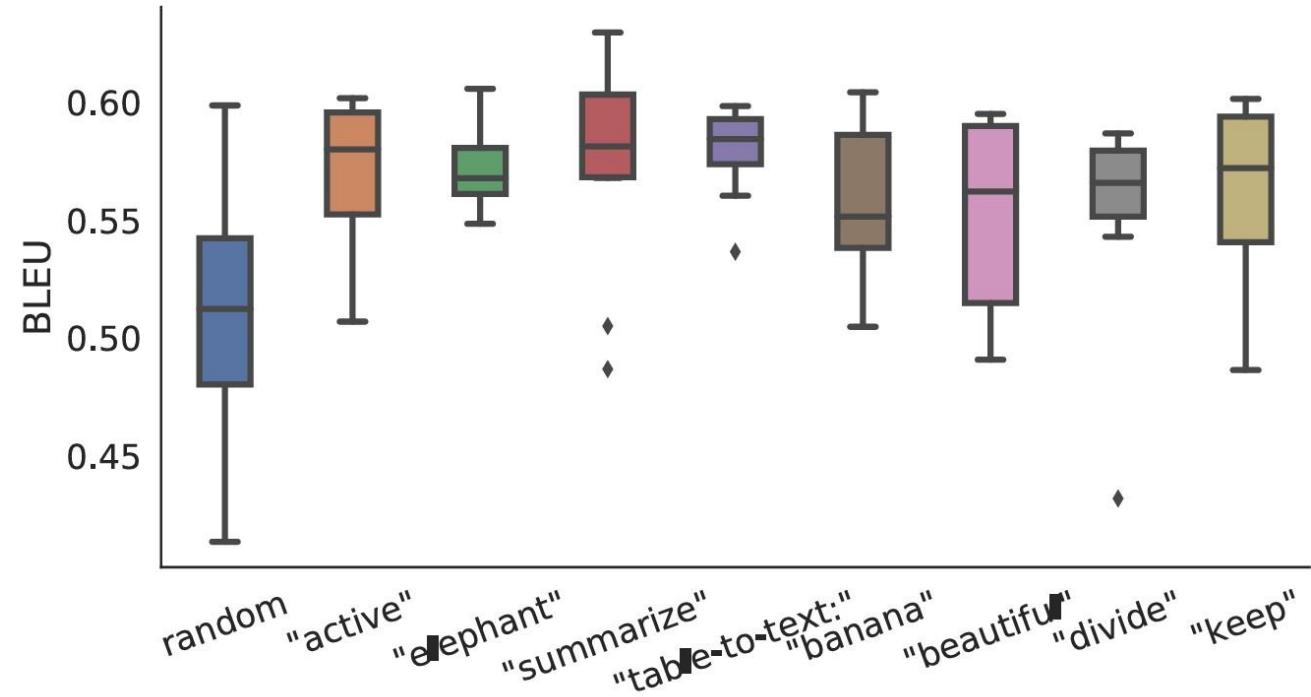


Figure 5: Initializing the prefix with activations of real words significantly outperforms random initialization, in low-data settings.

Issues with Prompt/Prefix-Tuning

- Optimal prefix length may be different for tasks
- The prefix occupies the length of your input context to the Transformer

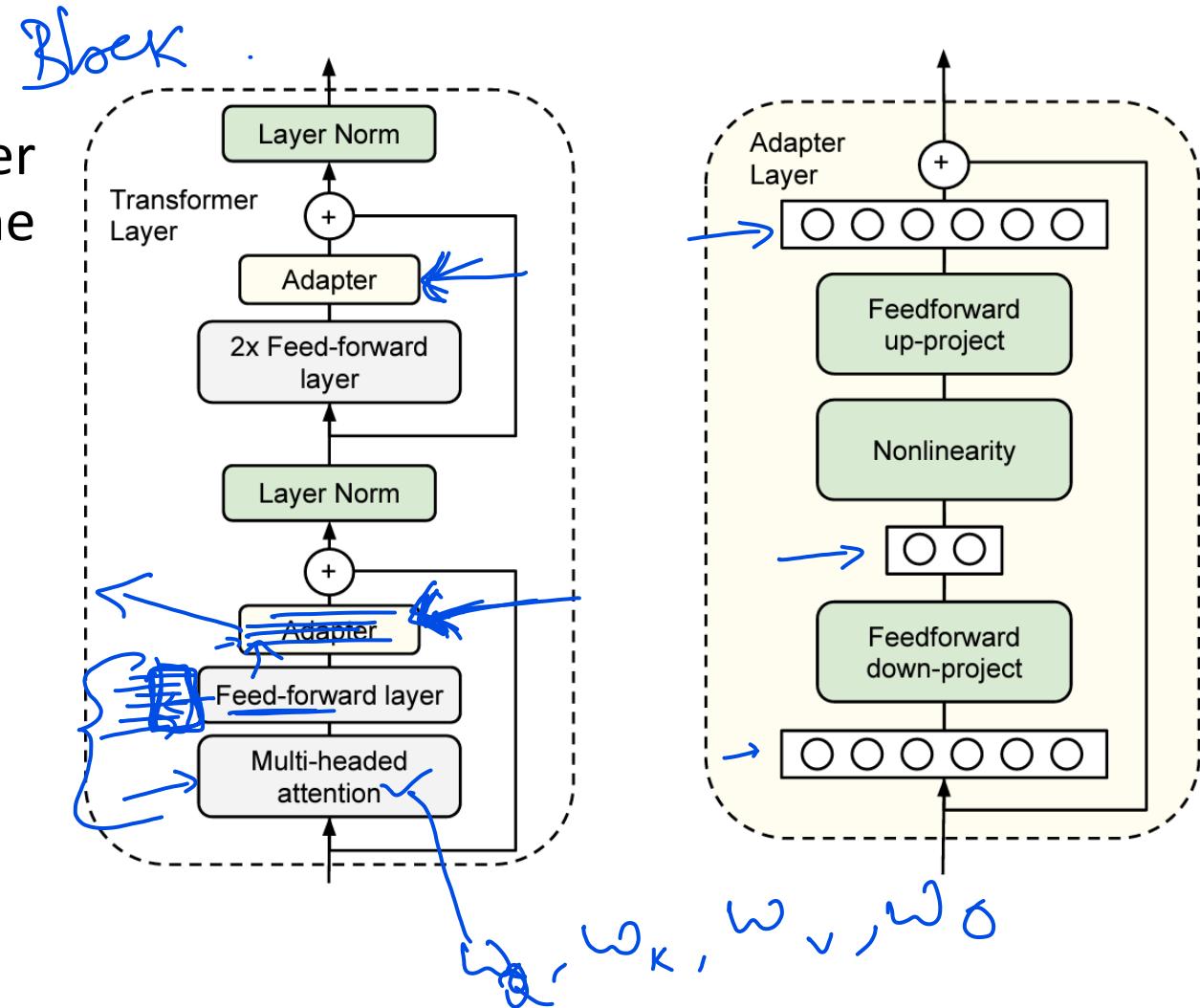
What are Adapters?

- Vanilla fine-tuning can be seen as adding an extra layer to the top of a Transformer
- Adapter modules perform more general architectural modifications: injecting new layers/modules into the original network.
- During training, the original network weights are untouched, only the adapter weights are updated.

Parameter-Efficient Transfer Learning for NLP

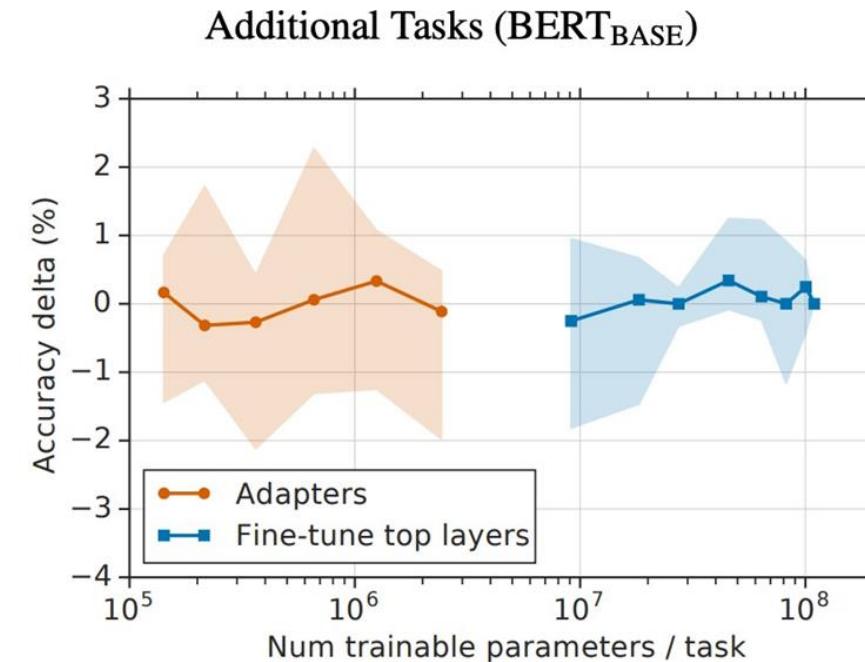
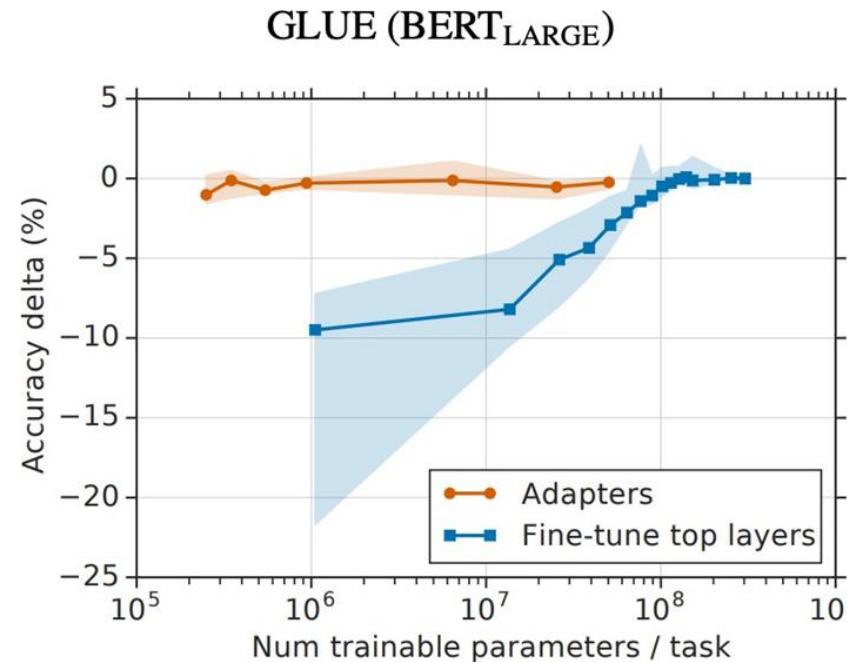
(Houlsby et. al, 2019)

- Adding adapter layers to each transformer layer: after the self-attention layer and the feed-forward layer
- Adapter modules have two main features
 - a small number of parameters
 - a near-identity initialization
- Only adapter layers and the final classification layer is updated during training



Comparison with Fine-Tuning

- Adapter-based tuning achieves a similar performance to full fine-tuning with several orders of magnitude fewer trained parameters.



Pros and Cons of Adapter-based Methods

- Pros:
 - Empirically very effective in multi-task settings
 - ✓ • Computationally efficient compared to full fine-tuning
- Cons:
 - ✓ • Adding in new layers makes the model slower during inference time
 - ✓ • Make the model size larger

Intrinsic Dimension

- An objective function's intrinsic dimension measures the minimum number of parameters needed to reach a satisfactory solution to the objective.
- Alternatively, the intrinsic dimension represents the lowest dimensional subspace in which one can optimize the original objective function to within a certain level of approximation error.

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Aghajanyan et. al, 2020.

Intrinsic Dimension

- Let θ^D be the parameters of a model
- Instead of optimizing θ^D , the subspace method optimizes θ^d in a lower dimensional space

$$\theta^D = \theta_0^D + P(\theta^d) \quad P : \mathbb{R}^d \rightarrow \mathbb{R}^D$$

- P is often a linear projection:

$$\theta^D = \theta_0^D + \theta^d M$$

- Fine-tuning tasks have a low intrinsic dimension: the number of parameters to be modified are several orders of magnitude less than the full parameterization of the pre-trained model.

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Aghajanyan et. al, 2020.

LoRA: Low-Rank Adaptation of Large Language Models

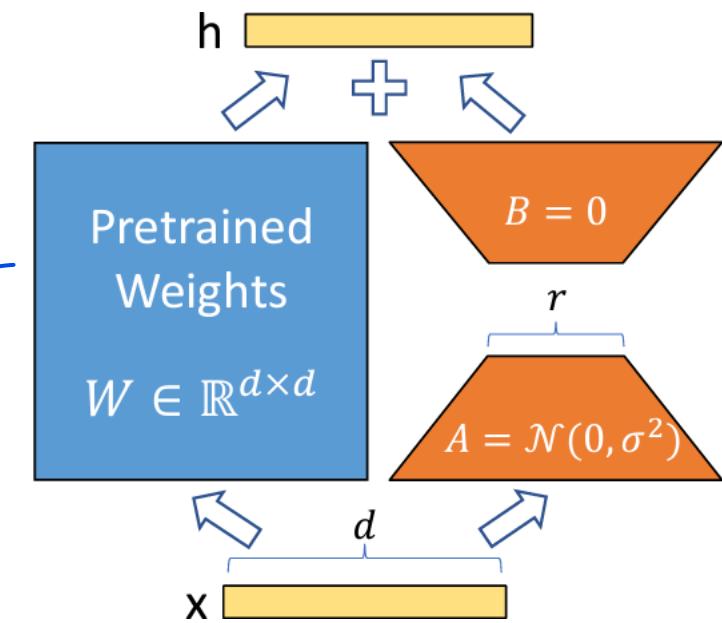
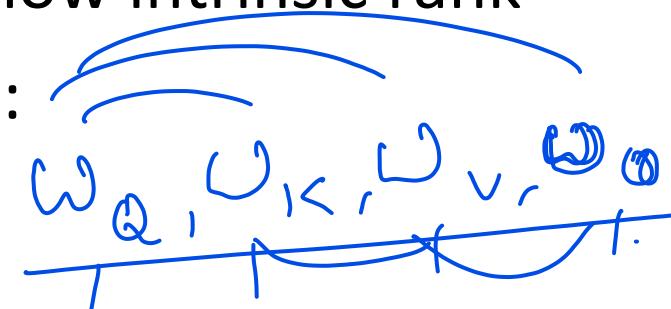
(Hu et. al, 2021)

- A neural network contains many dense layers which perform matrix multiplication.
- Inspired by the low intrinsic dimension assumption, hypothesize that the update weights can also have a low intrinsic rank
- Pre-trained matrix to be updated:

$$W_0 \in \mathbb{R}^{d \times k}$$

- Updated matrix

$$W_0 + \Delta W$$



LoRA: Low-Rank Adaptation of Large Language Models

(Hu et. al, 2021)

- Reparametrize the updated weight with low-rank decomposition

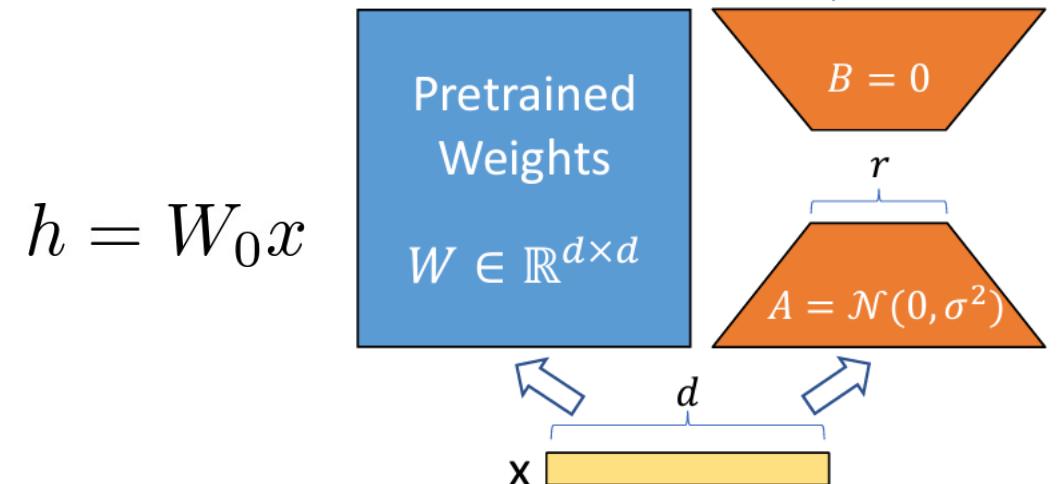
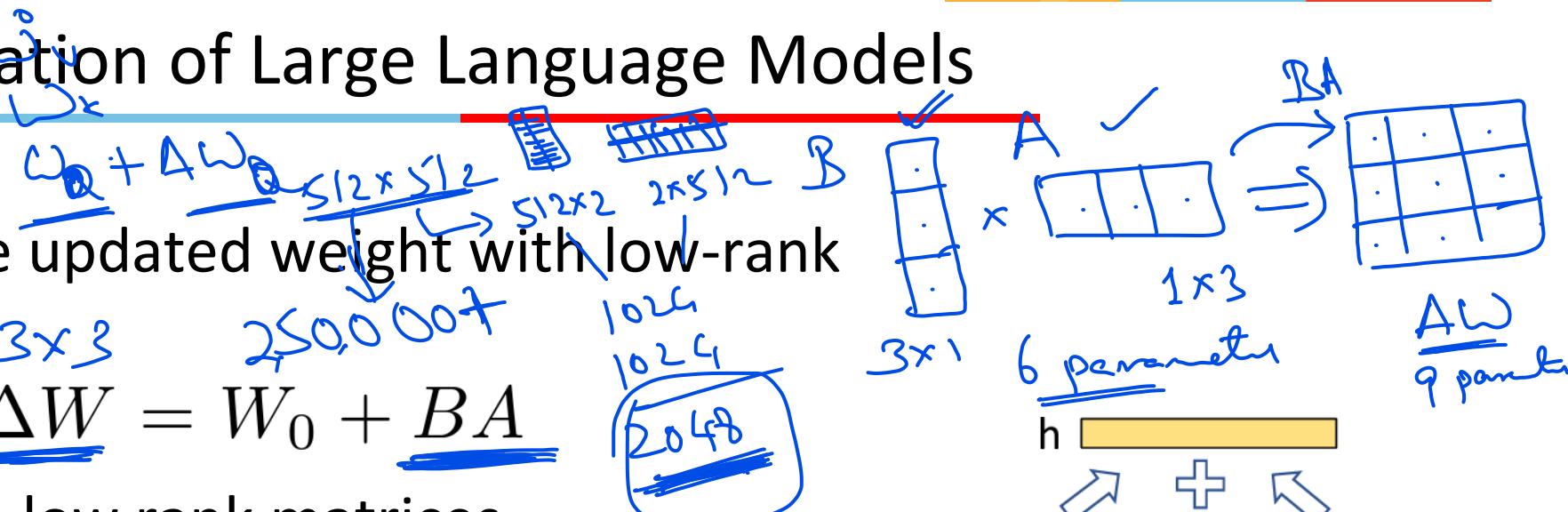
$$W_0 + \Delta W = W_0 + BA$$

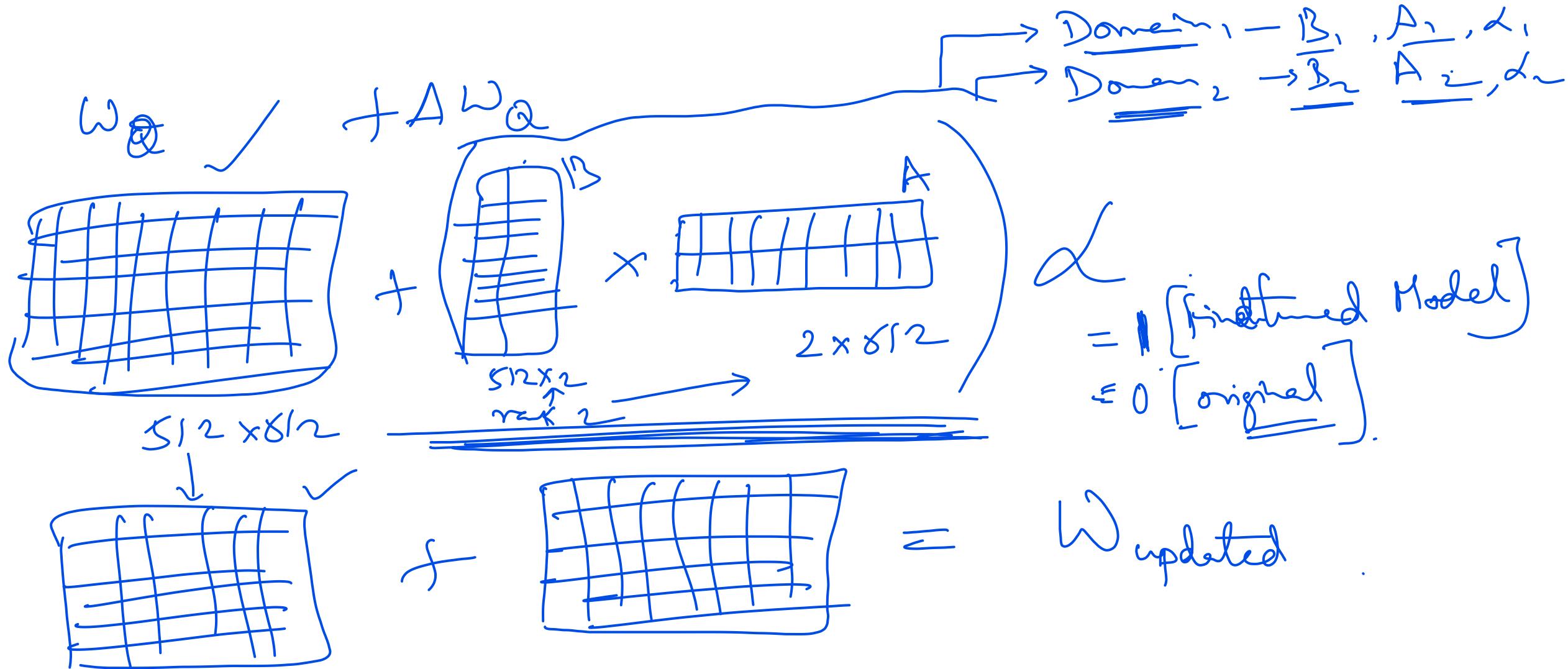
- where A and B are low rank matrices

$$A \in \mathbb{R}^{r \times k} \quad B \in \mathbb{R}^{d \times r}$$

- For the hidden state h of an input x ,
- The updated hidden state is now

$$h = W_0x + \Delta Wx = W_0x + BAx$$





Applying LoRA to Transformers

- In principle, LoRA can be applied to any weight matrices in deep learning
- In this study, they focus on applying LoRA to attention matrices in Transformers
 - r ranges from 2 to 64
 - For GPT3-175B
 - VRAM: 1.2TB → 350GB
 - Checkpoint storage: 350GB → 35MB (10000x smaller)

Comparison with Other Fine-Tuning Methods

Model&Method	# Trainable Parameters	WikiSQL	MNLI-m	SAMSum
		Acc. (%)	Acc. (%)	R1/R2/RL
GPT-3 (FT)	175,255.8M	73.8	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter ^H)	7.1M	71.9	<u>89.8</u>	53.0/28.9/44.8
GPT-3 (Adapter ^H)	40.1M	73.2	91.5	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	91.7	53.8/29.8/45.9
GPT-3 (LoRA)	37.7M	74.0	91.6	53.4/29.2/45.1

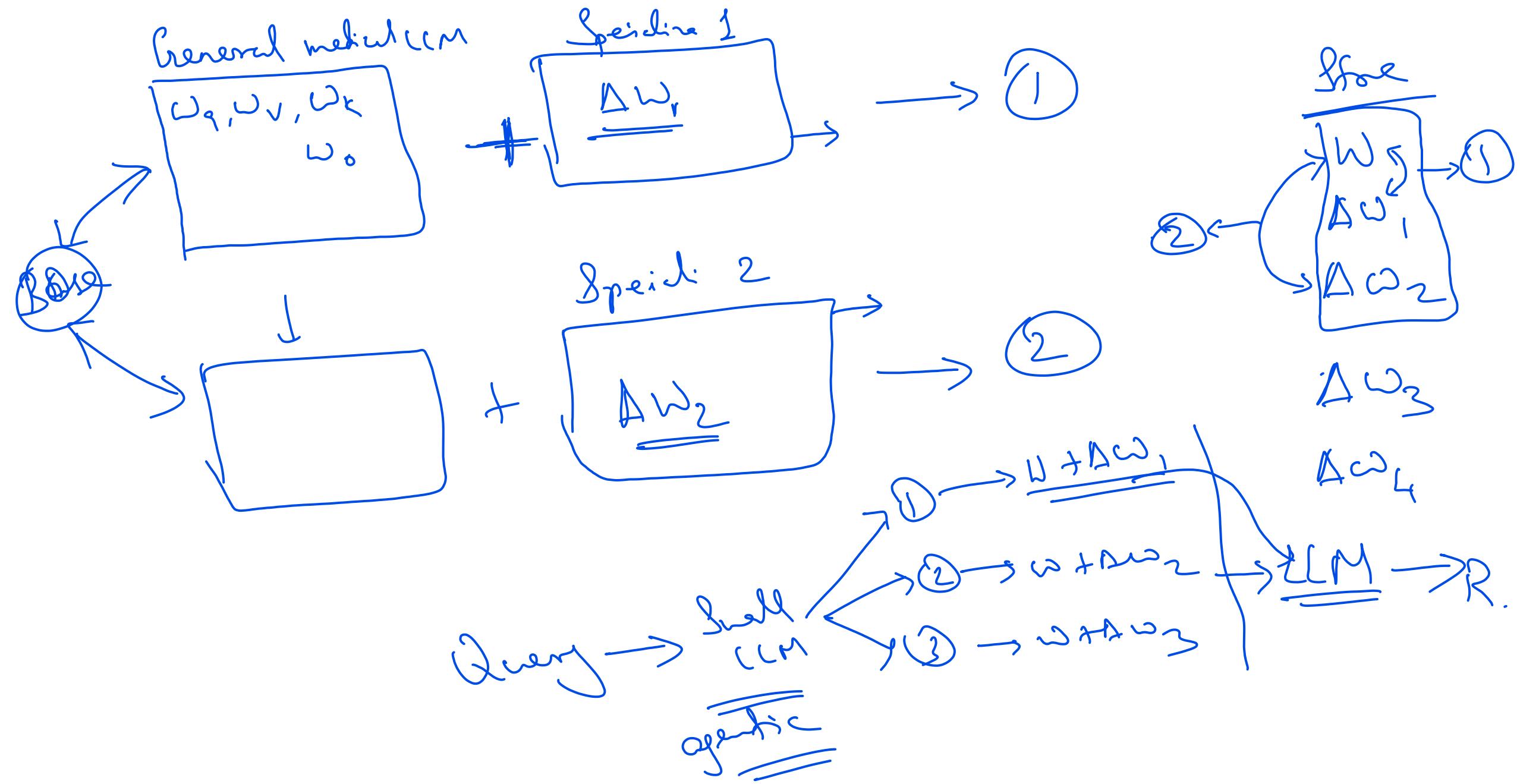
- LoRA outperforms several baselines with comparable or fewer trainable parameters.

Which Matrices Should We Apply LoRA to?

$\underline{W_q + \Delta W_q}$
 $\underline{\cancel{W_v + \Delta W_v}}$

		# of Trainable Parameters = 18M						
Weight Type		W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r		8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)		70.4	70.0	73.0	73.2	71.4	73.7	73.7
MultiNLI ($\pm 0.1\%$)		91.0	90.8	91.0	91.3	91.3	91.3	91.7

- Putting all the parameters in ΔW_q or ΔW_k lower performance, while adapting both result. W_q W_v
 results in significantly and yields a good



Optimal Rank for LoRA

- $r = 4$ and $r = 8$ already give a good result, and increasing r does not cover more meaningful subspaces

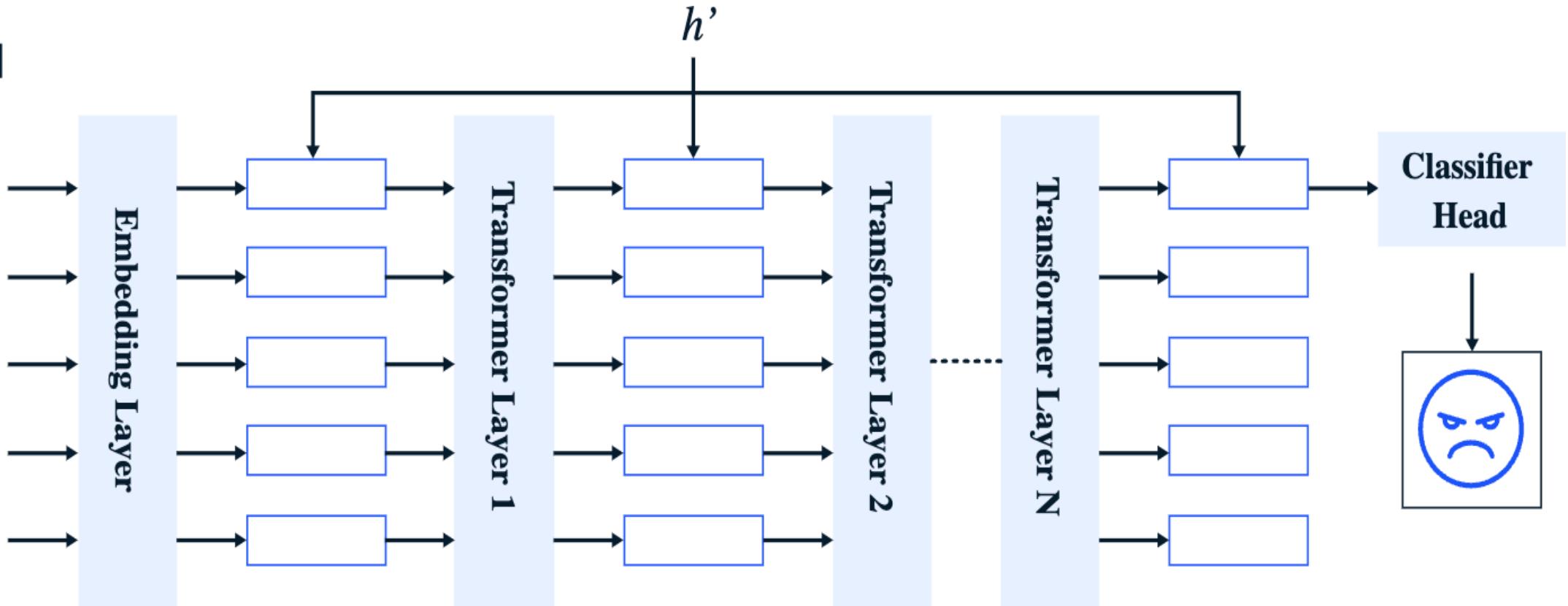
	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Summary of Parameter-Efficient Fine-Tuning

- Vanilla Fine-Tuning

[CLS]

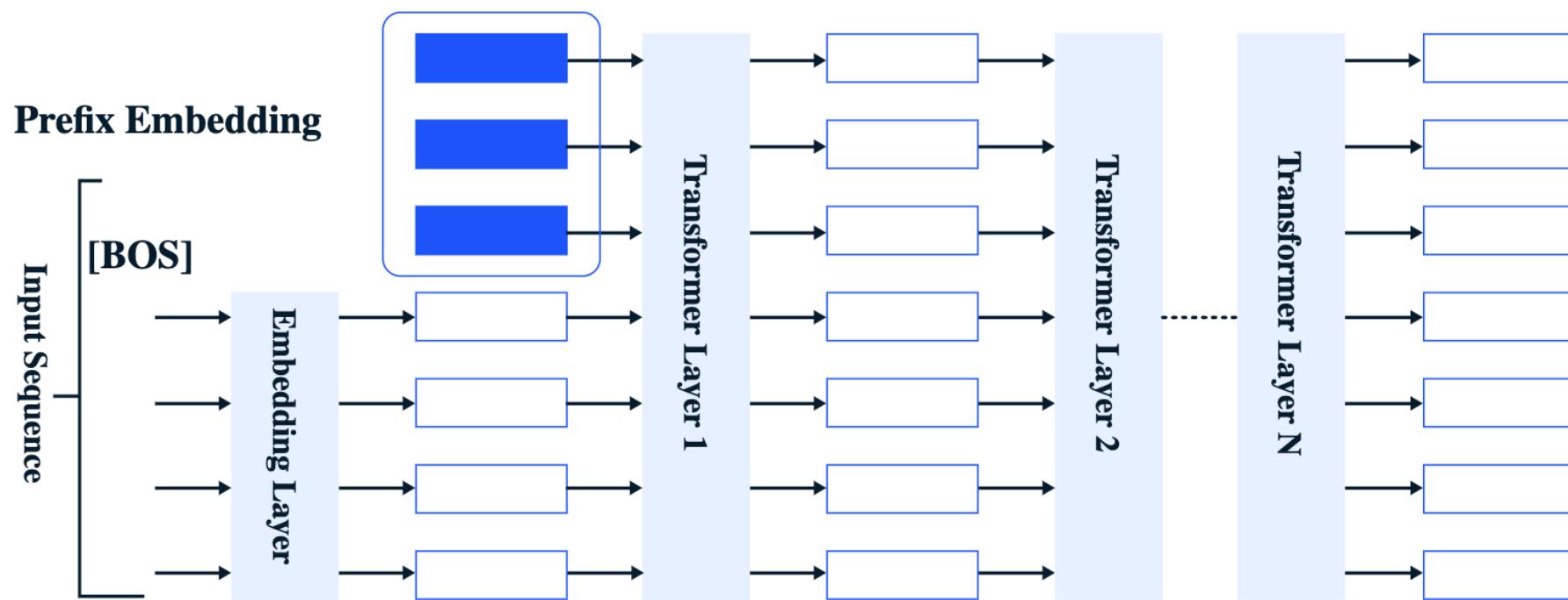
This is a total waste of money



Source: <https://www.leewayhertz.com/parameter-efficient-fine-tuning/>

Summary of Parameter-Efficient Fine-Tuning

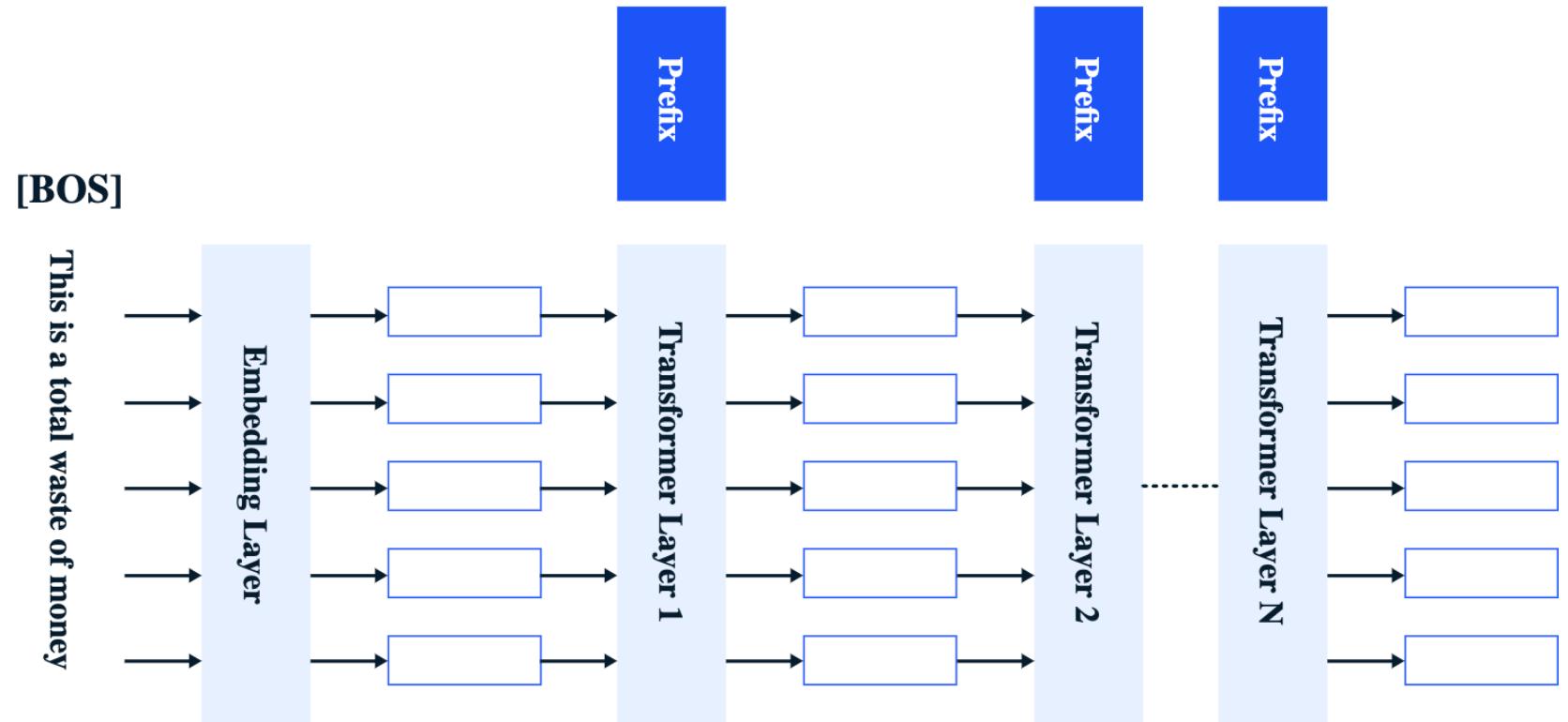
- Prompt-Tuning



Source: <https://www.leewayhertz.com/parameter-efficient-fine-tuning/>

Summary of Parameter-Efficient Fine-Tuning

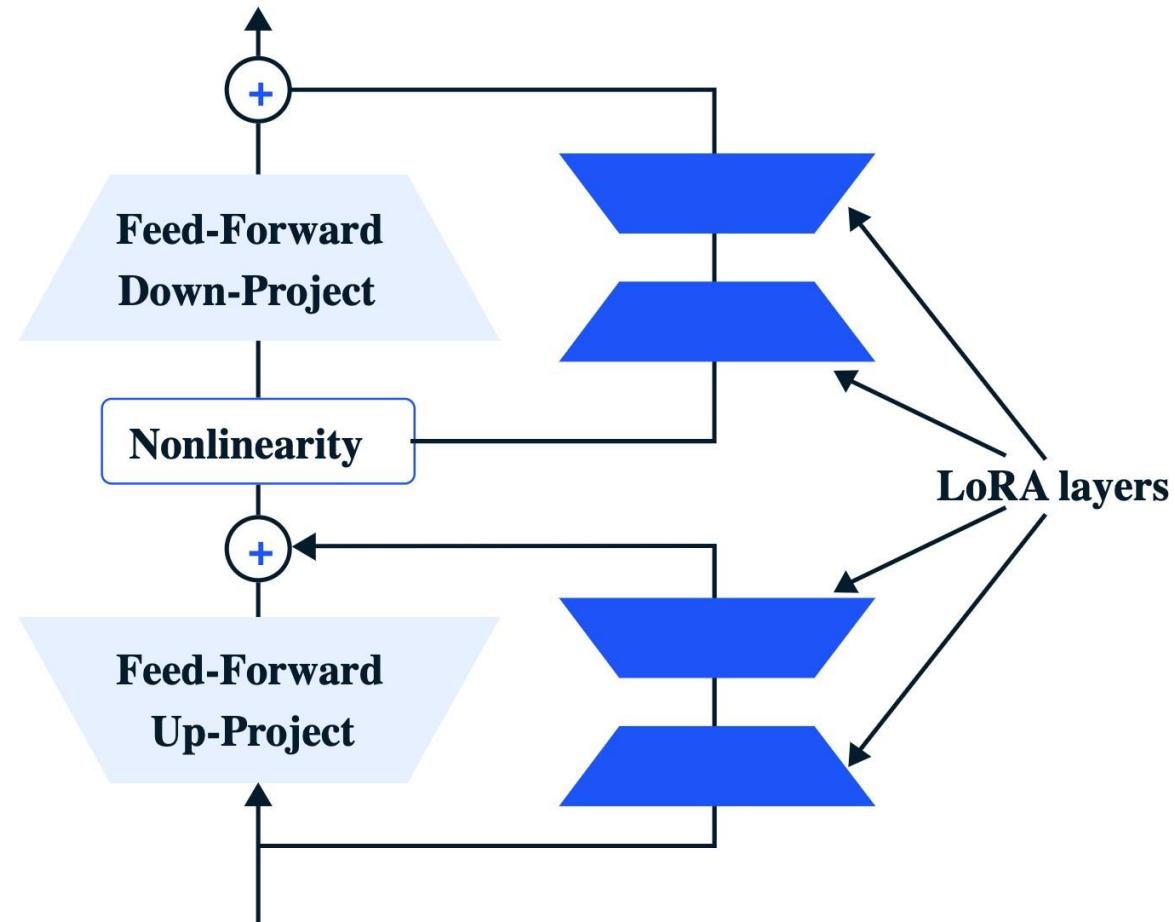
- Prefix-Tuning



Source: <https://www.leewayhertz.com/parameter-efficient-fine-tuning/>

Summary of Parameter-Efficient Fine-Tuning

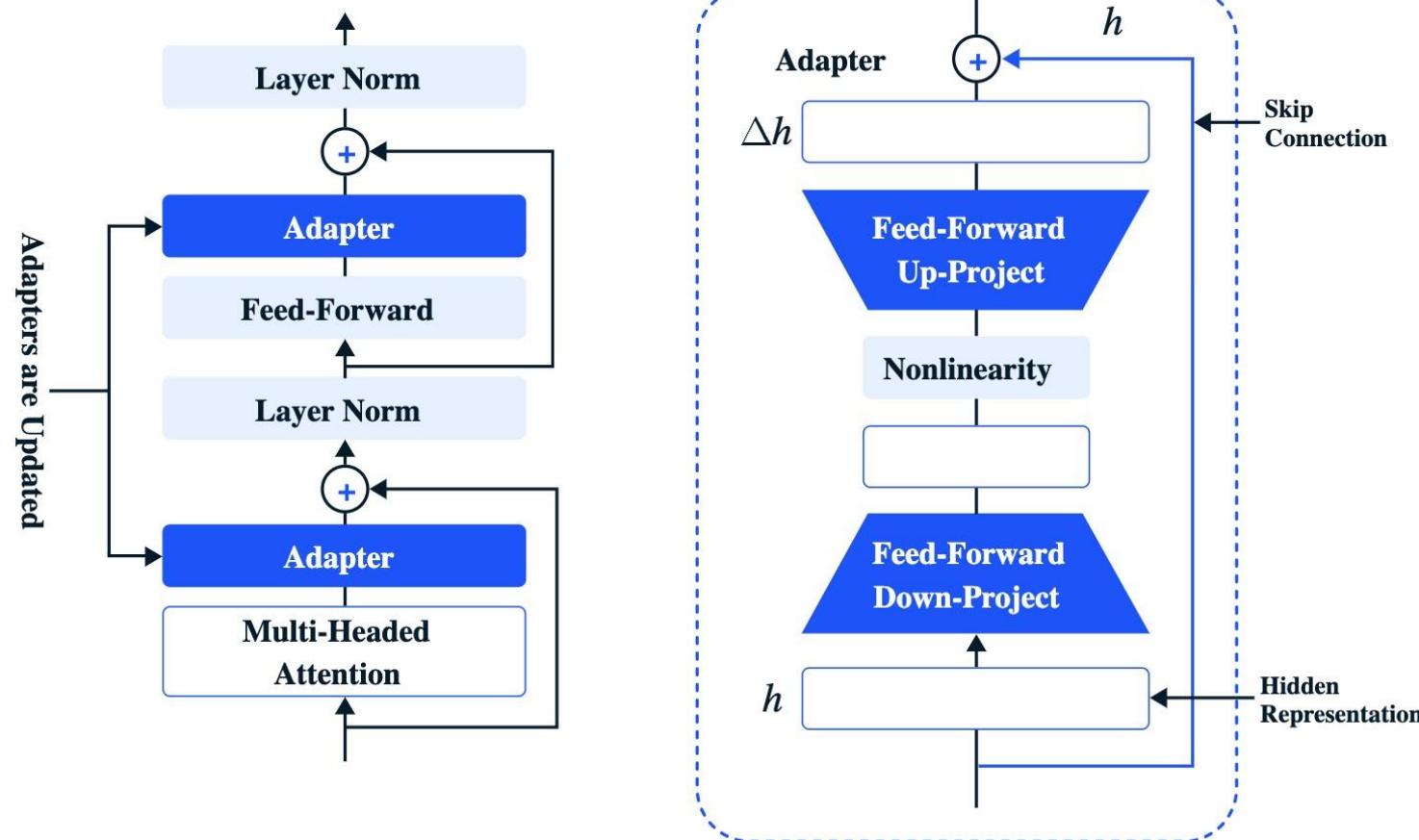
- LoRA



Source: <https://www.leewayhertz.com/parameter-efficient-fine-tuning/>

Summary of Parameter-Efficient Fine-Tuning

- Adapters

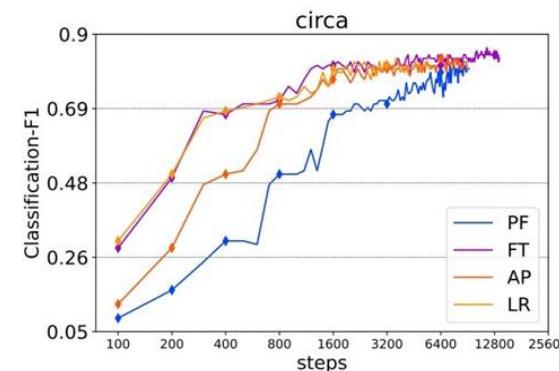
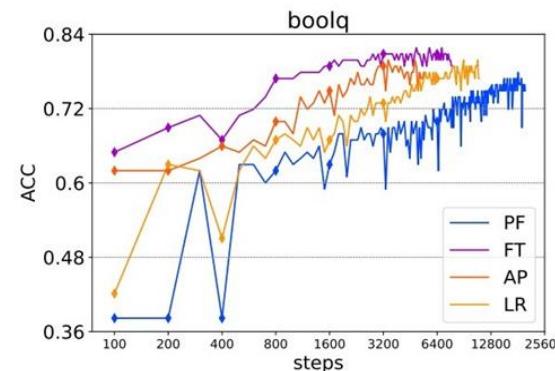
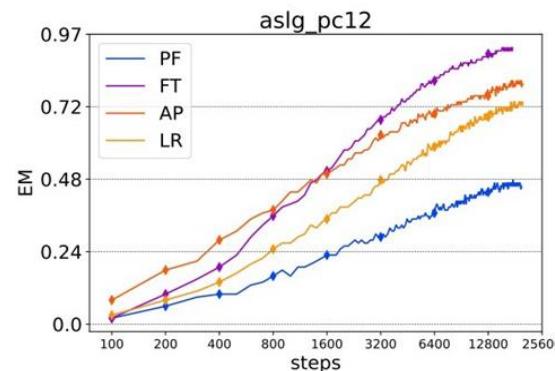
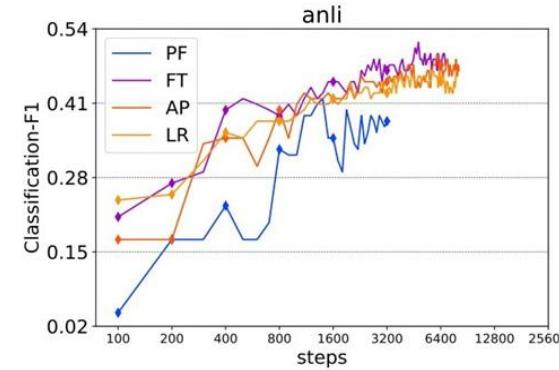
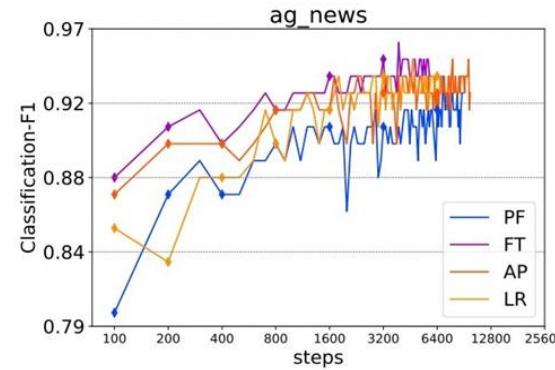
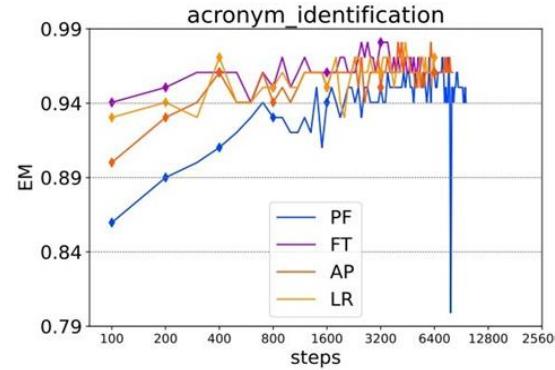


Source: <https://www.leewayhertz.com/parameter-efficient-fine-tuning/>

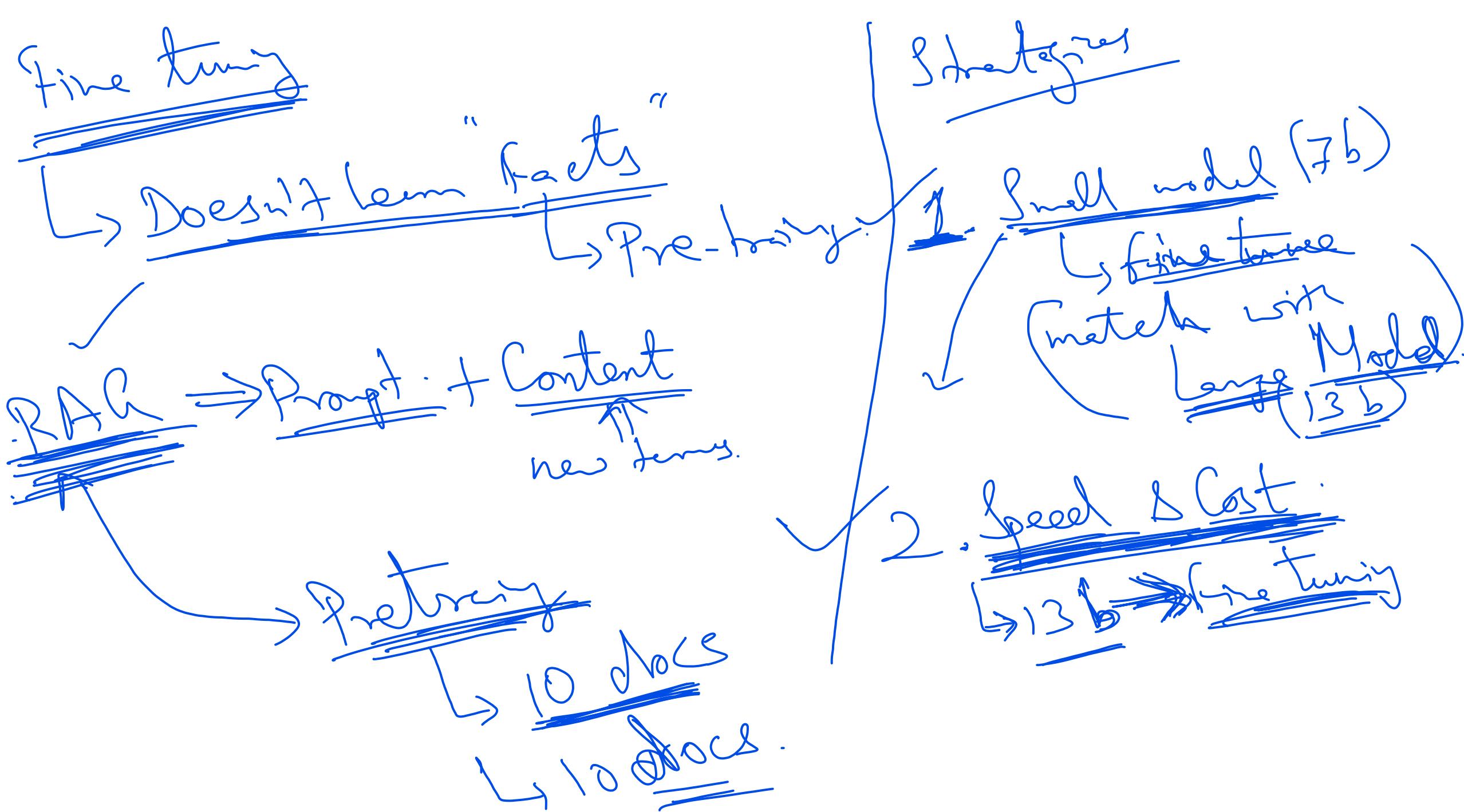
Performance Comparison on Various NLP Tasks

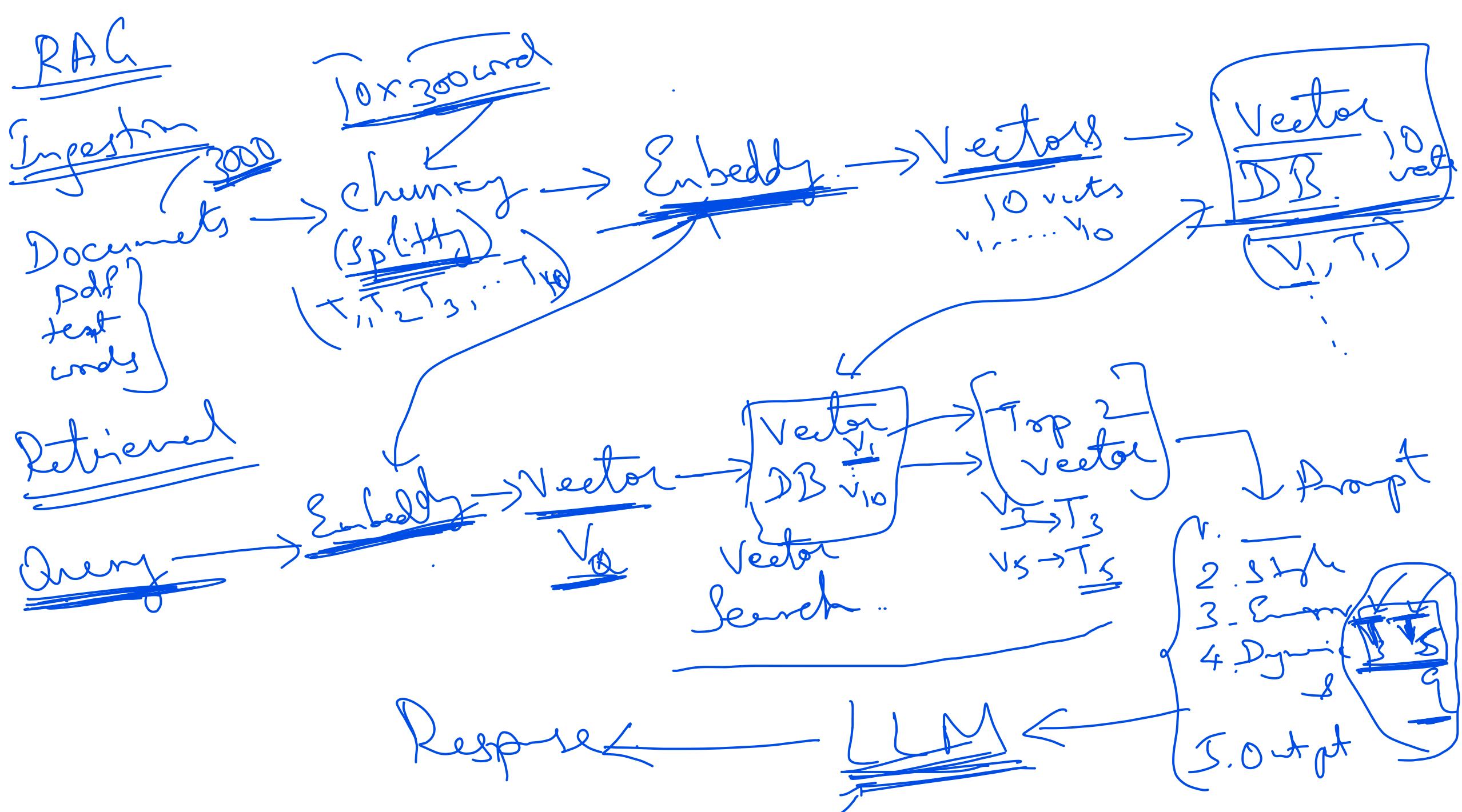
- If you have enough data and computing resources:
- Overall performance (on T5-base):

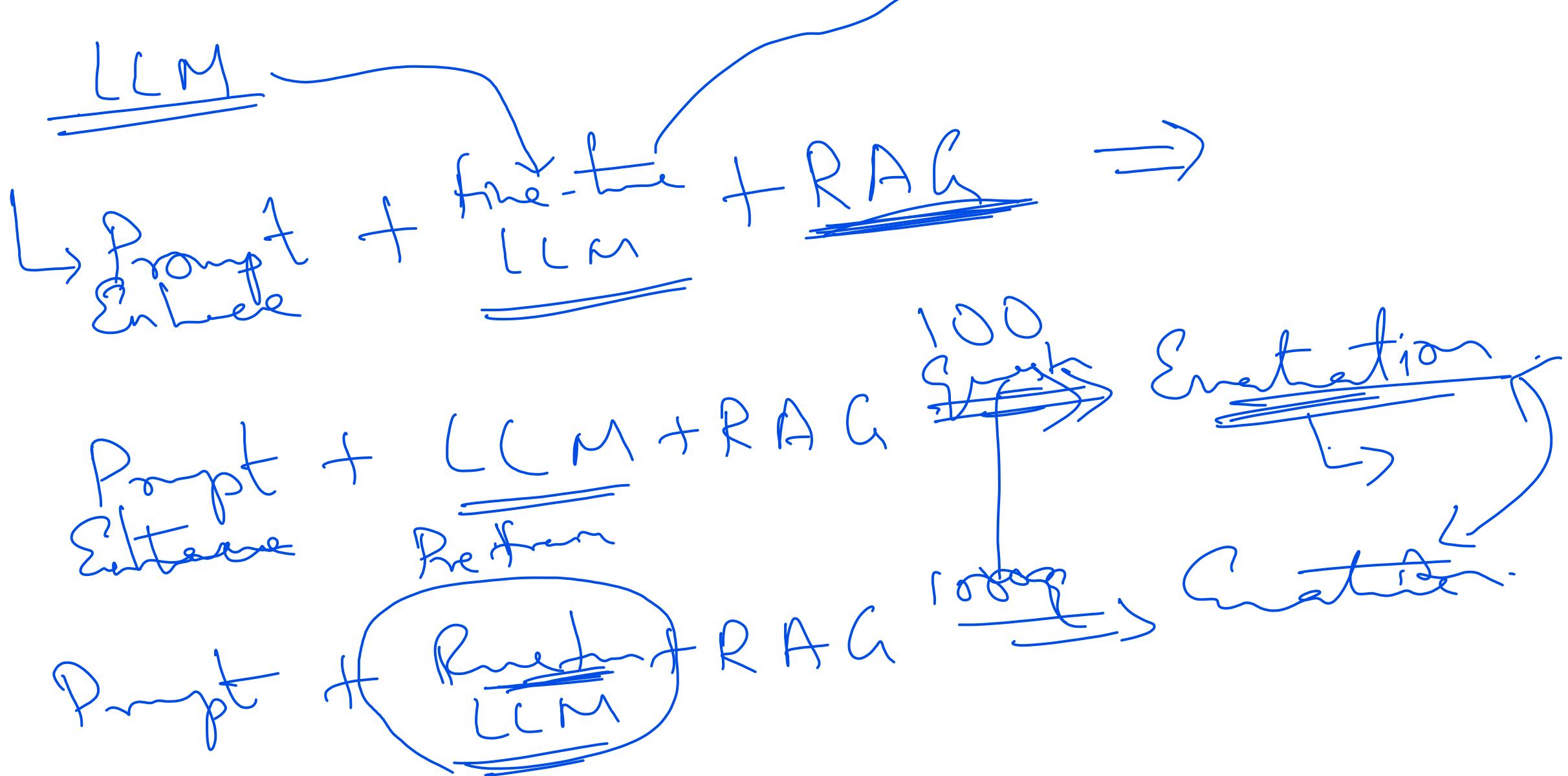
Full fine-tuning > LoRA > Adapters > Prefix Tuning > Prompt Tuning



Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models (Ding et. al, 2023)







Till now

1. LLM / Transformer - Pretrain

2. Fine tuning (LORA) - Domain Sp.

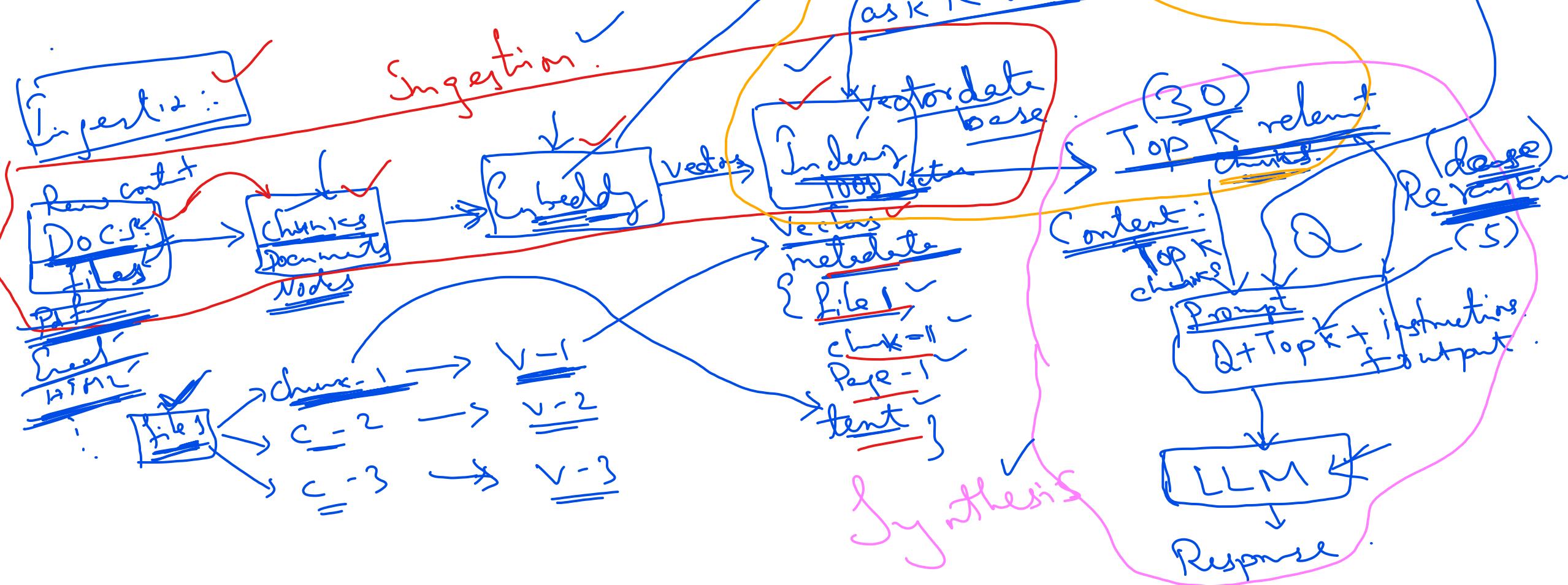
Challenges:-

- 1. Data (used for tuning) has to be updated.
- 2. Computational ~~Expensive~~ is efficient.
- 3. Hallucinate (rephrase).
"Black box" →
- 4. Prompt. Engg. is also insufficient.

RAC

3 components :-

1. Ingestion
2. Retrieval
3. Synthesis



chunking:-

Text file :-

1000 lines, 300 sentences

→ Spitters

cheat sheet
Sentences (Paraparag)

Semantic

Markdown

Section wise

IO-files

chunking: techniques

→ IO-file
small to big →

256

1024

1B

IO-file

Hierarchical

doc → FC-1 → FC-2 → FC-3 → FC-4

→ C-1 → C-2 → C-3 → C-4

RAPTOR

→ C-5 → C-6

Q1: am bhagat & I belong
in baghout work for XYZ
Eijaz is travel agent.
I drive car.

Q2: who works for
XYZ comp?

bhagat

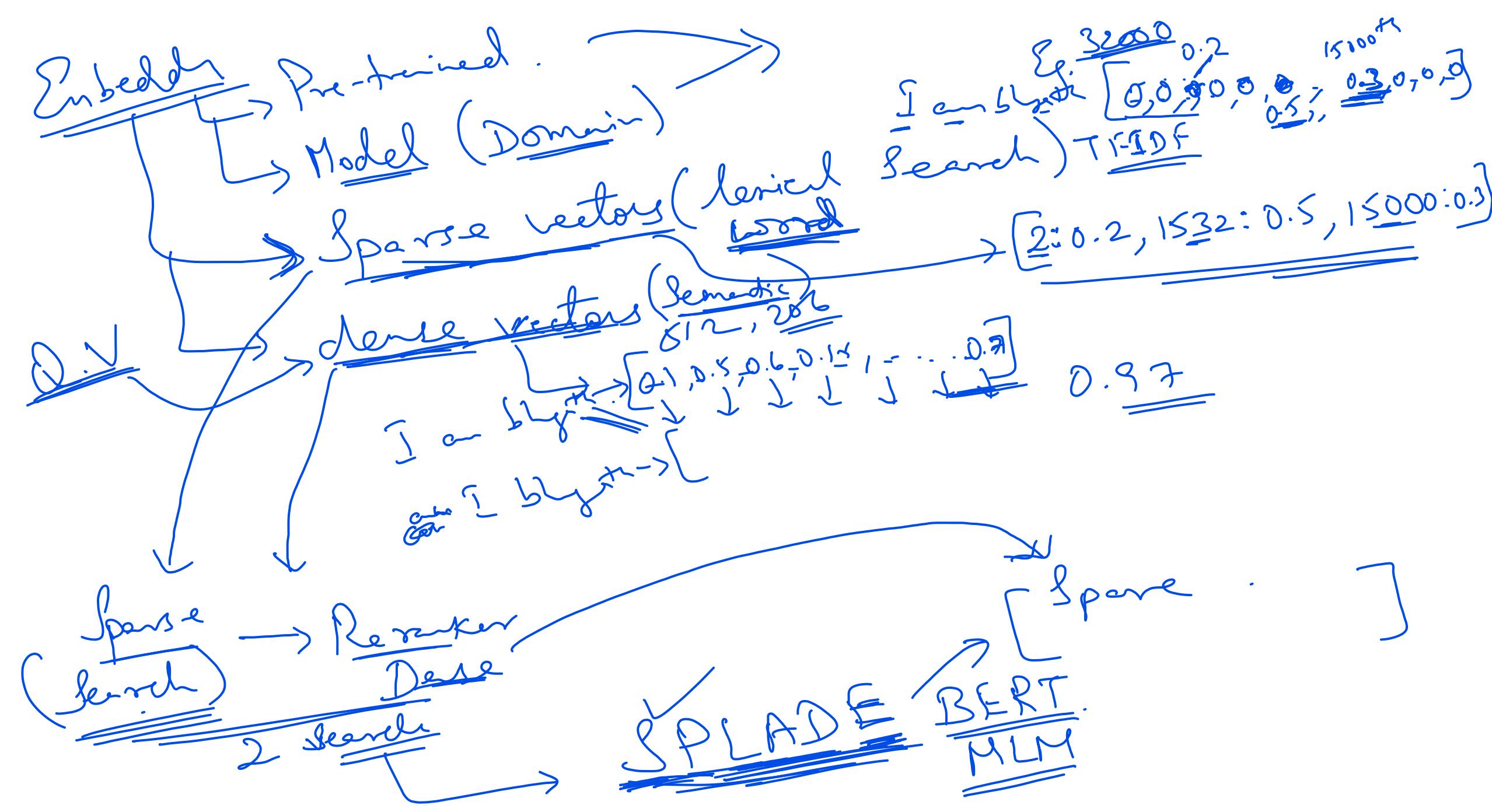
Vector base

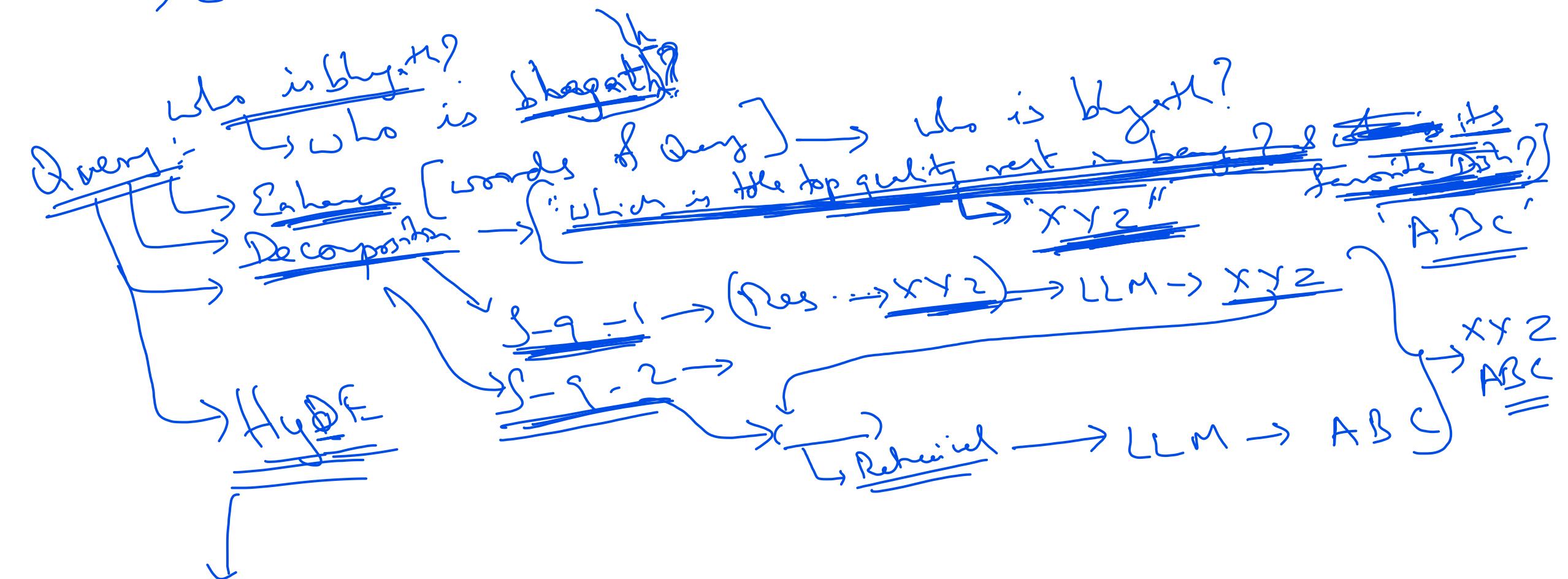
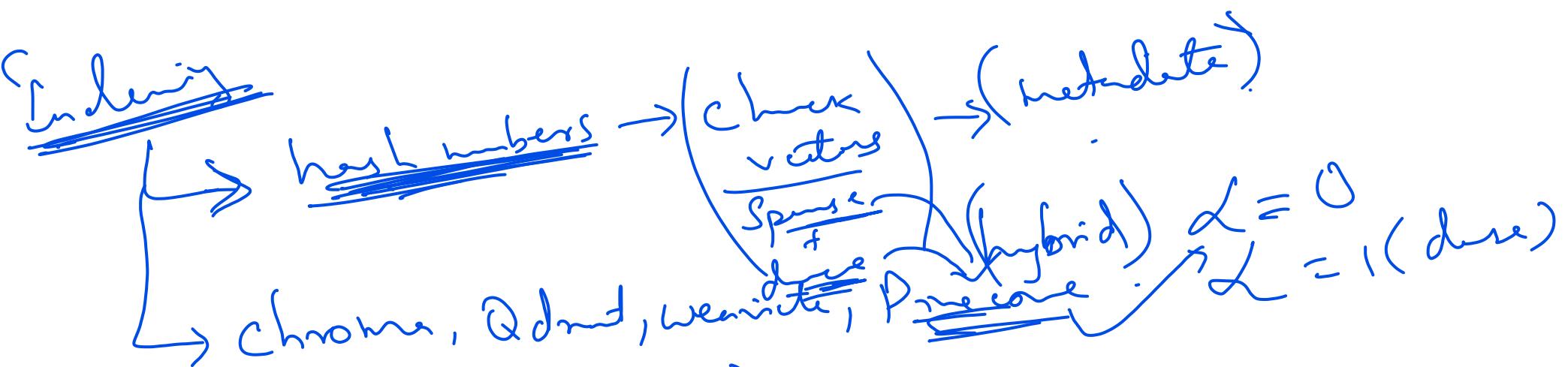
ch-1

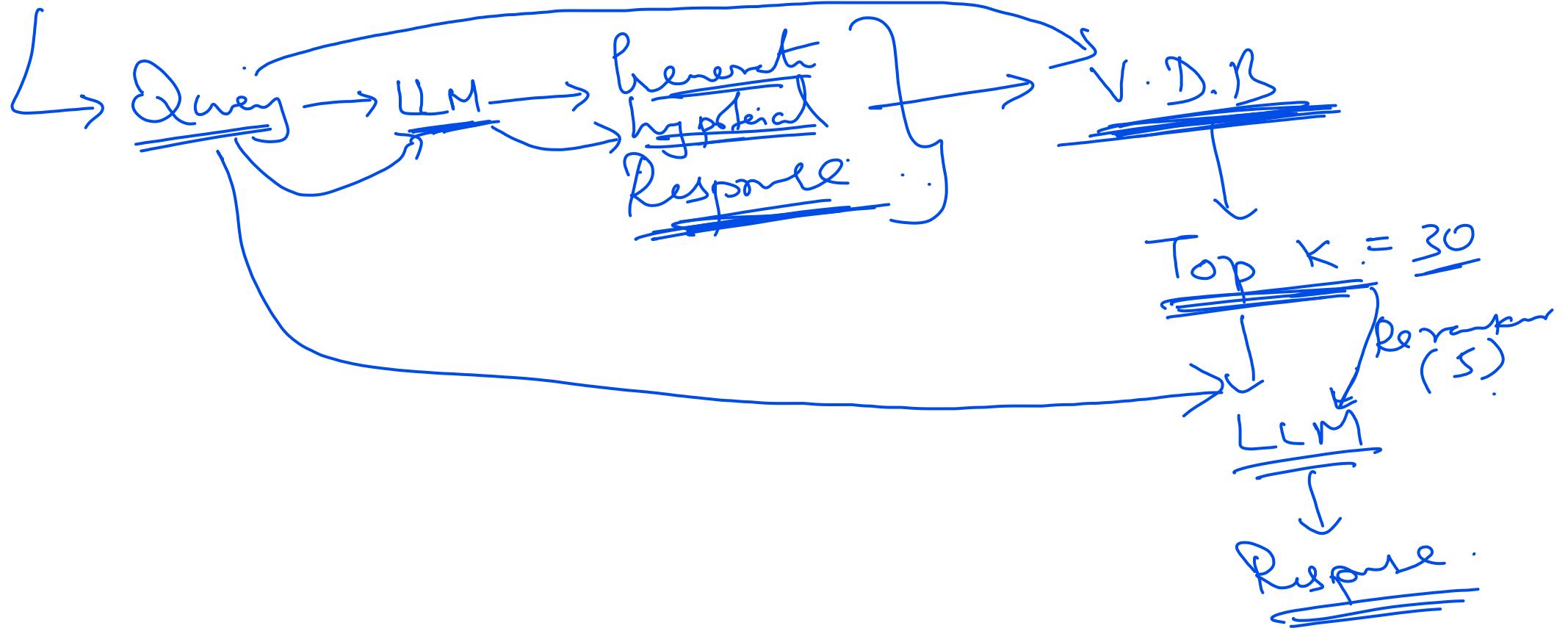
ch-2

xyz
work
agent
travel
agent

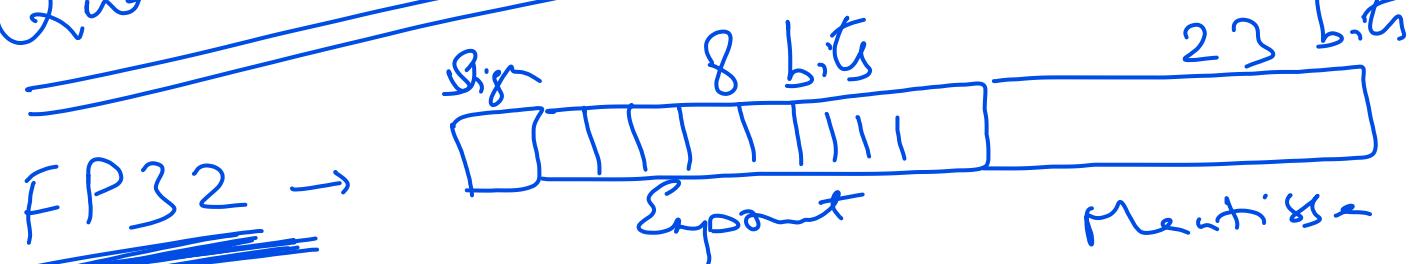
LLM







Quantization:-



Uma $\xrightarrow{70b}$ FP32 → 4-bit 40GB → 40GB.

-4-5:1
2-3:1

8+4

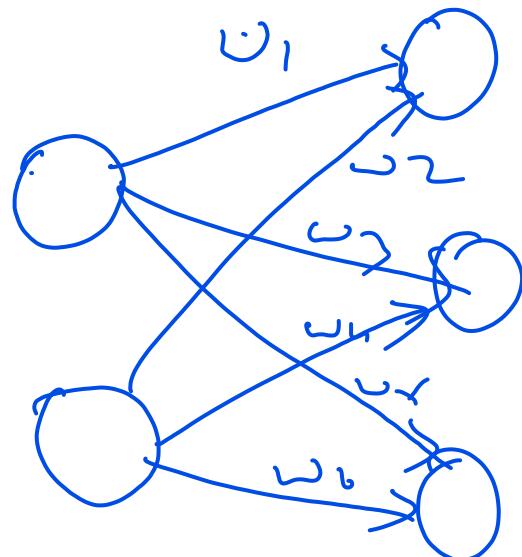
1-bit LLM $\xrightarrow{1.58}$ bit $[-1, 0, 1]$ → $G = \begin{bmatrix} 0.7, 0.6, -0.6, 0.2 \\ 1, 1, -1, 0 \end{bmatrix}$

$\omega = \begin{bmatrix} 0.7, 0.6, -0.6, 0.2 \\ 1, 1, -1, 0 \end{bmatrix}$

$\omega = \begin{bmatrix} 1, 1, -1, 0 \end{bmatrix}$
for $L \rightarrow (-0.5 \text{ to } 0.5)$

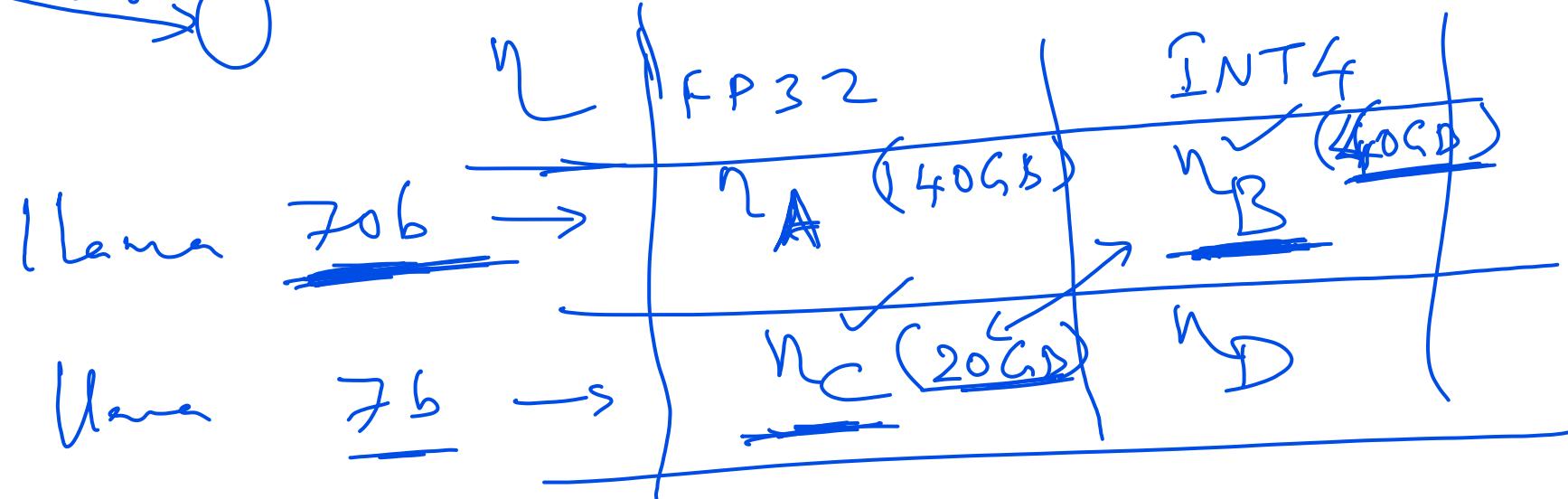
large LLM → Uma 3-7 bits → FP32 → bit
more efficient

Small Uma 3-7 b.
FP32



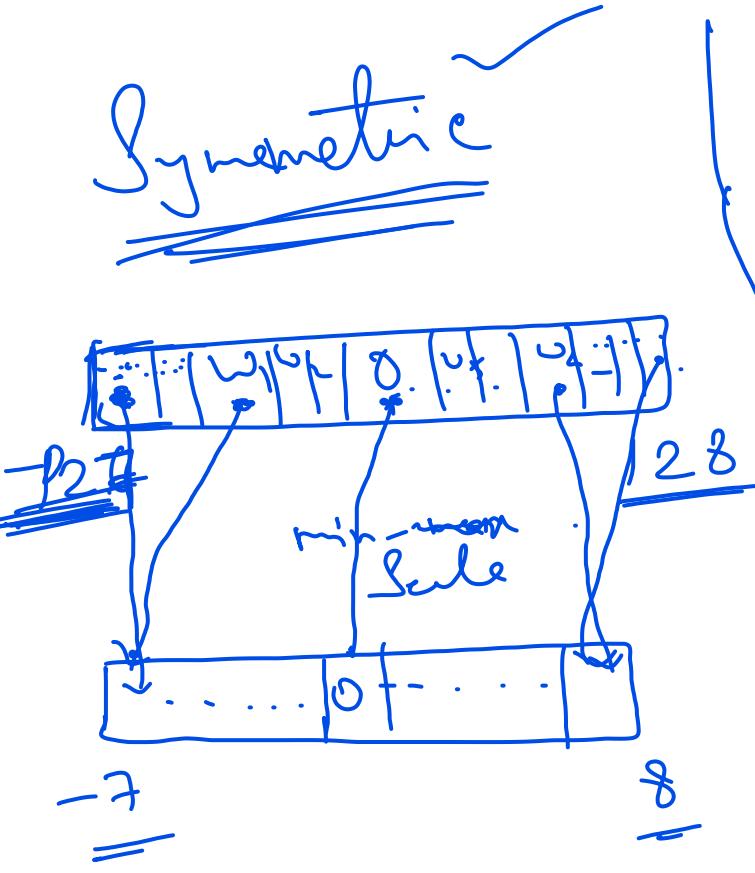
$$w_1 = \frac{1}{1, 2, 3, 6, 7, 8, 2} \quad (\text{by sum})$$

$$w_1 = 1 \quad (\text{Int 4}) \rightarrow \text{less sum}$$

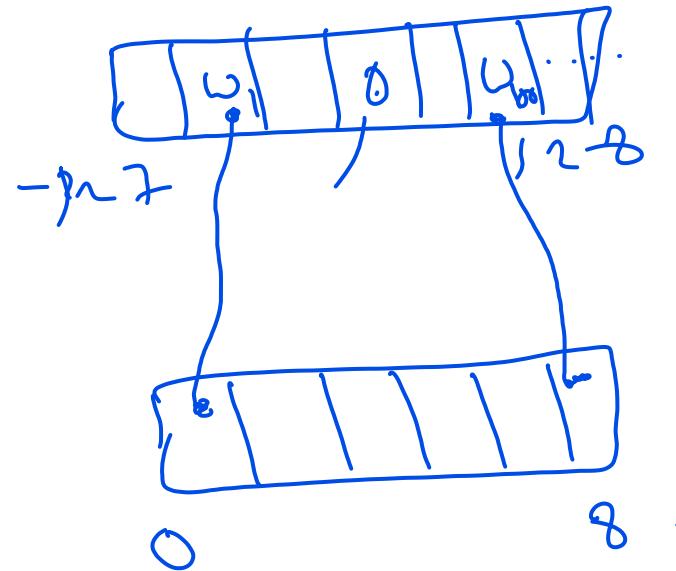


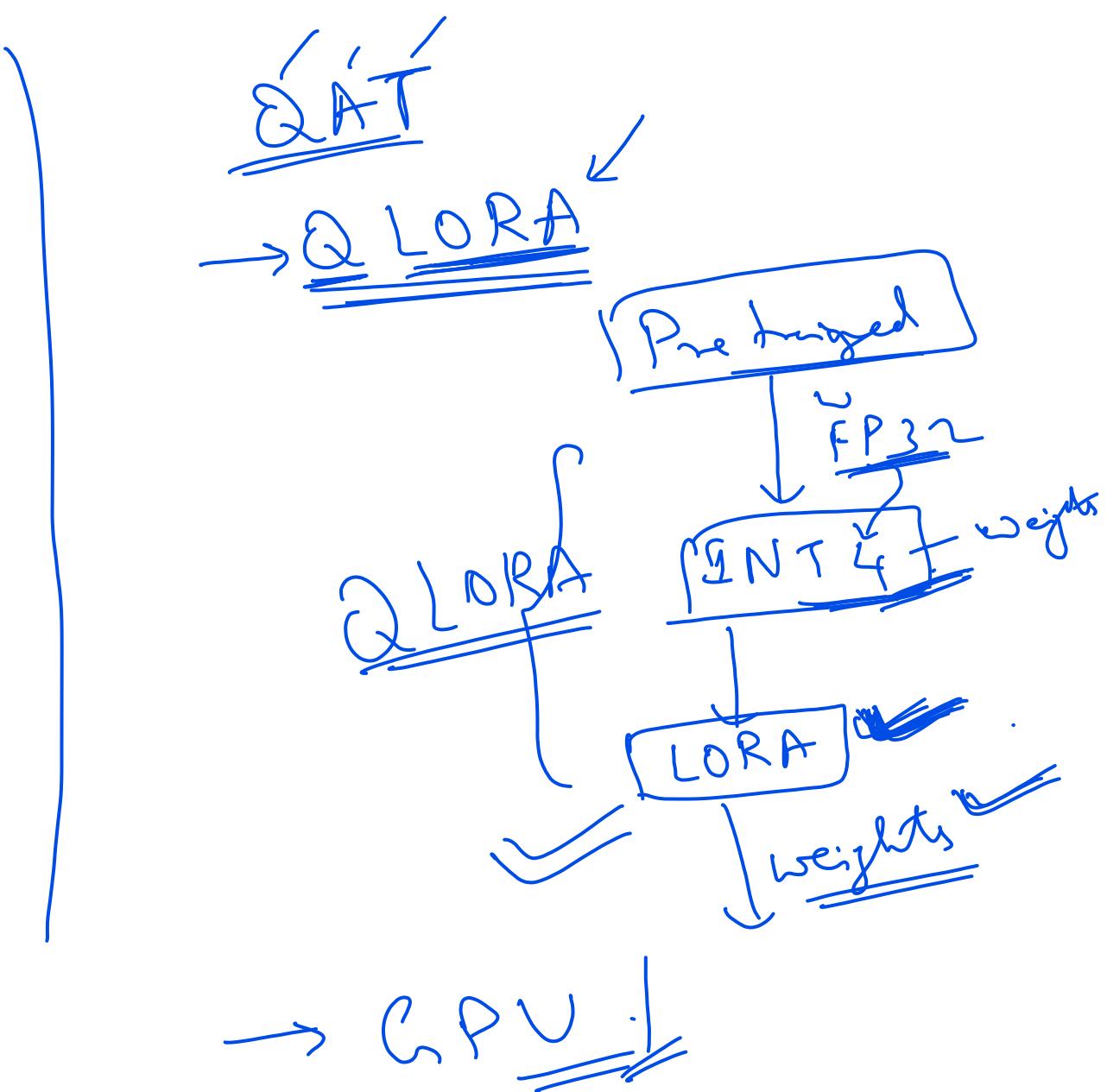
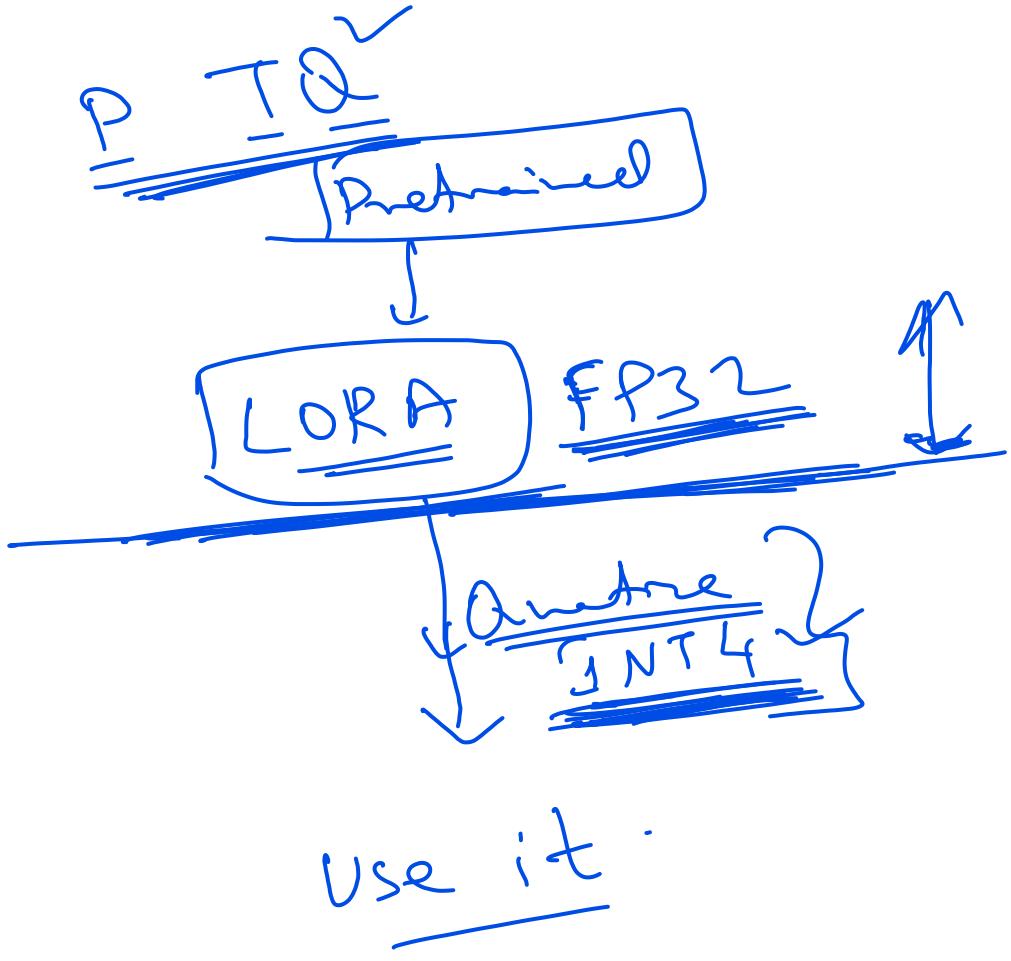
$$n_B > n_C$$

Symmetric



Asymetric





Next class

→ 14th July - 8 pm to 10 pm.

→ UX Conversations

→ Revision → Post topics → Answers [Plid Exam]

12 marks → MCQ { }
12 marks → Subjective { }

week4_RAG.md

[Week 4] Retrieval Augmented Generation

🔗

ETMI5: Explain to Me in 5

🔗

In this week's content, we will do an in-depth exploration of Retrieval Augmented Generation (RAG), an AI framework that enhances the capabilities of Large Language Models by integrating real-time, contextually relevant information from external sources during the response generation process. It addresses the limitations of LLMs, such as inconsistency and lack of domain-specific knowledge, hence reducing the risk of generating incorrect or hallucinated responses.

RAG operates in three key phases: ingestion, retrieval, and synthesis. In the ingestion phase, documents are segmented into smaller, manageable chunks, which are then transformed into embeddings and stored in an index for efficient retrieval. The retrieval phase involves leveraging the index to retrieve the top-k relevant documents based on similarity metrics when a user query is received. Finally, in the synthesis phase, the LLM utilizes the retrieved information along with its internal training data to formulate accurate responses to user queries.

We will discuss the history of RAG and then delve into the key components of RAG, including ingestion, retrieval, and synthesis, providing detailed insights into each phase's processes and strategies for improvement. We will also go over various challenges associated with RAG, such as data ingestion complexity, efficient embedding, and fine-tuning for generalization and propose solutions to each of them.

What is RAG? (Recap)

🔗

Retrieval Augmented Generation (RAG) is an AI framework that enhances the quality of responses generated by LLMs by incorporating up-to-date and contextually relevant information from external sources during the generation process. It addresses the inconsistency and lack of domain-specific knowledge in LLMs, reducing the chances of hallucinations or incorrect responses. RAG involves two phases: retrieval, where relevant information is searched and retrieved, and content generation, where the LLM synthesizes an answer based on the retrieved information and its internal training data. This approach improves accuracy, allows source verification, and reduces the need for continuous model retraining.

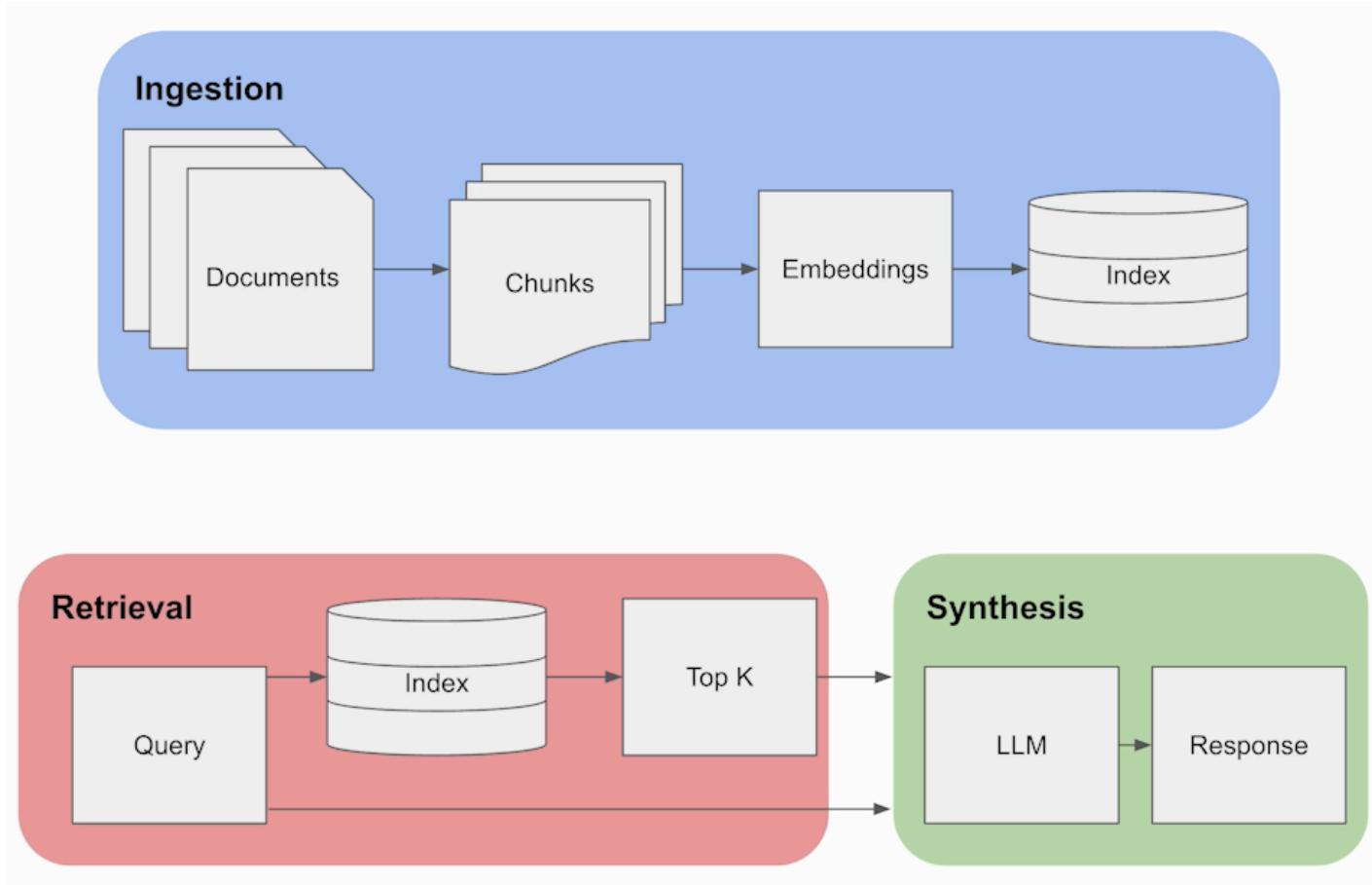


Image Source: <https://www.deeplearning.ai/short-courses/langchain-for-lm-application-development/>

The diagram above outlines the fundamental RAG pipeline, consisting of three key components:

1. Ingestion:

- Documents undergo segmentation into chunks, and embeddings are generated from these chunks, subsequently stored in an index.
- Chunks are essential for pinpointing the relevant information in response to a given query, resembling a standard retrieval approach.

2. Retrieval:

- Leveraging the index of embeddings, the system retrieves the top-k documents when a query is received, based on the similarity of embeddings.

3. Synthesis:

- Examining the chunks as contextual information, the LLM utilizes this knowledge to formulate accurate responses.

💡 Unlike previous methods for domain adaptation, it's important to highlight that RAG doesn't necessitate any model training whatsoever. It can be readily applied without the need for training when specific domain data is provided.

History



RAG, or Retrieval-Augmented Generation, made its debut in [this](#) paper by Meta. The idea came about in response to the limitations observed in large pre-trained language models regarding their ability to access and manipulate knowledge effectively.

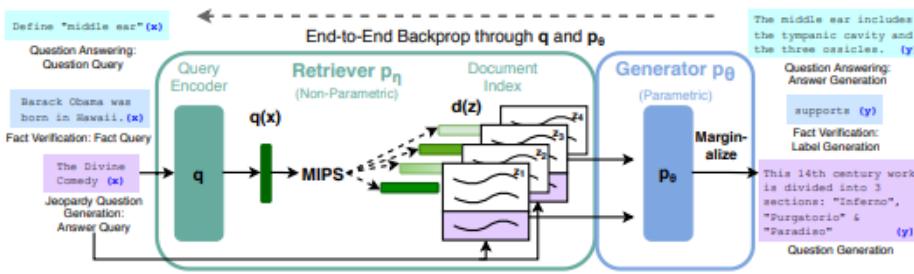


Image Source: https://arxiv.org/pdf/2005.11401.pdf

Below is a short summary of how the authors introduce the problem and provide a solution:

RAG came about because, even though big language models were good at remembering facts and performing specific tasks, they struggled when it came to precisely using and manipulating that knowledge. This became evident in tasks heavy on knowledge, where other specialized models outperformed them. The authors identified challenges in existing models, such as difficulty explaining decisions and keeping up with real-world changes. Before RAG, there were promising results with hybrid models that mixed both parametric and non-parametric memories. Examples like REALM and ORQA combined masked language models with a retriever, showing positive outcomes in this direction.

Then, along came RAG, a game-changer in the form of a flexible fine-tuning method for retrieval-augmented generation. RAG combined pre-trained parametric memory (like a seq2seq model) with non-parametric memory from a dense vector index of Wikipedia, accessed through a pre-trained neural retriever like Dense Passage Retriever (DPR). RAG models aimed to enhance pre-trained, parametric-memory generation models by combining them with non-parametric memory through fine-tuning. The seq2seq model in RAG used latent documents retrieved by the neural retriever, creating a model trained end-to-end. Training involved fine-tuning on any seq2seq task, learning both the generator and retriever. Latent documents were then handled using a top-K approximation, either per output or per token.

RAG's main significance was moving away from past approaches that proposed adding non-parametric memory to systems. Instead, RAG explored a new approach where both parametric and non-parametric memory components were pre-trained and filled with lots of knowledge. In experiments, RAG proved its worth by achieving top-notch results in open-domain question answering and surpassing previous models in fact verification and knowledge-intensive generation. Another win for RAG was showing it could adapt, allowing the non-parametric memory to be swapped out and updated to keep the model's knowledge fresh in a changing world.

Key Components

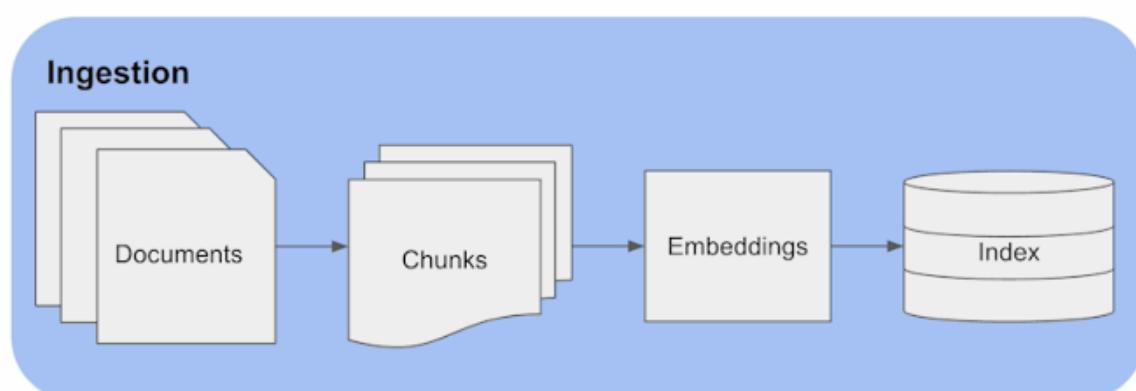
②

As mentioned earlier, the key elements of RAG involve the processes of ingestion, retrieval, and synthesis. Now, let's delve deeper into each of these components.

Ingestion

②

In RAG, the ingestion process refers to the handling and preparation of data before it is utilized by the model for generating responses.



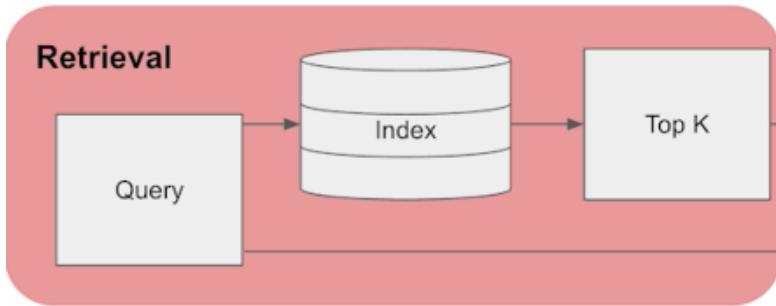
This process involves 3 key steps:

1. **Chunking:** Breaking down input text into smaller, more manageable segments or chunks. This can be based on size, sentences, or other natural divisions within the text. We will dig deeper into chunking strategies in the next sections. As an example, consider a comprehensive article on the Renaissance. The chunking process involves breaking down the article into manageable segments based on natural breaks, such as paragraphs or distinct historical periods (e.g., Early Renaissance, High Renaissance). Each of these segments becomes a chunk, enabling focused analysis by the language model.
2. **Embedding:** Transforming the text or chunks into a vector format that captures essential qualities in a computationally friendly way. This step is crucial for efficient processing by the language model. Following from the previous example—once the article segments are identified, the embedding process transforms the content of each chunk into a vector format. For instance, the section on the High Renaissance could be embedded into a vector that captures key artistic, cultural, and historical aspects. This vector representation enhances the model's ability to understand and process the nuanced information within the chunk.
3. **Indexing:** Organizing the embedded data in a structured format optimized for quick and efficient retrieval. This often involves creating a vector representation for each document and storing these vectors in a searchable format, such as a vector database or search engine. In the example we discussed—the indexed database is created by organizing these vector representations of historical events. Each chunk, now represented as a vector, is indexed for efficient retrieval. When a user queries about a specific aspect of the Renaissance, the indexing enables the quick identification and retrieval of the most relevant chunks, providing contextually rich responses.

Retrieval



The retrieval component involves the following steps:



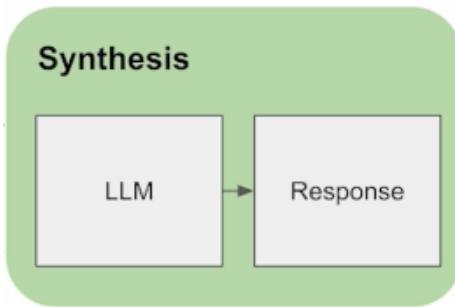
1. **User Query:** A user poses a natural language query to the LLM. For instance, let's say we've completed the ingestion process for renaissance articles as explained in the above method and a user poses a query, "Tell me about the Renaissance period."
2. **Query Conversion:** The query is sent to an embedding model, which converts the natural language query into a numeric format, creating an embedding or vector representation. The embedding model is the same as the model used to embed articles in the ingestion phase.
3. **Vector Comparison:** The numeric vectors of the query are compared to vectors in an index of a knowledge base created in the previous phase. This involves measuring similarity or distance metrics between the query vector and vectors stored in the index (often cosine similarity).
4. **Top-K Retrieval:** The system then retrieves the top-K documents or passages from the knowledge base that have the highest similarity to the query vector. This step involves selecting a predefined number (K) of the most relevant documents based on the vector similarities. These embeddings may include information about different aspects of the Renaissance.
5. **Data Retrieval:** The system retrieves the actual content or data from the selected top-K documents in the knowledge base. This content is typically in human-readable form, representing relevant information related to the user's query.

Therefore, at the end of the retrieval phase, the LLM has access to relevant context regarding the segments of the knowledge base that hold utmost relevance to the user's query. In this example, the retrieval process ensures that the user receives a well-informed response about the Renaissance, drawing on historical documents stored in the knowledge base to provide contextually rich information.

Synthesis



The Synthesis phase is very similar to regular LLM generation, except that now the LLM has access to additional context from the knowledge base. The LLM presents the final answer to the user, combining its own language generation with information retrieved from the knowledge base. The response may include references to specific documents or historical sources.



RAG Challenges



Although RAG seems to be a very straightforward way to integrate LLMs with knowledge, there are still the below mentioned open research and application challenges with RAG.

- Data Ingestion Complexity:** Dealing with the complexity of ingesting extensive knowledge bases involves overcoming engineering challenges. For instance, parallelizing requests effectively, managing retry mechanisms, and scaling infrastructure are critical considerations. Imagine ingesting large volumes of diverse data sources, such as scientific articles, and ensuring efficient processing for subsequent retrieval and generation tasks.
- Efficient Embedding:** Ensuring the efficient embedding of large datasets poses challenges like addressing rate limits, implementing robust retry logic, and managing self-hosted models. Consider the scenario where an AI system needs to embed a vast collection of news articles, requiring strategies to handle changing data, syncing mechanisms, and optimizing embedding costs.
- Vector Database Considerations:** Storing data in a vector database introduces considerations such as understanding compute resources, monitoring, sharding, and addressing potential bottlenecks. Think about the challenges involved in maintaining a vector database for a diverse range of documents, each with varying levels of complexity and importance.
- Fine-Tuning and Generalization:** Fine-tuning RAG models for specific tasks while ensuring generalization across diverse knowledge-intensive NLP tasks is challenging. For instance, achieving optimal performance in question-answering tasks might require different fine-tuning approaches compared to tasks involving creative language generation, requiring careful balance.
- Hybrid Parametric and Non-Parametric Memory:** Integrating parametric and non-parametric memory components in models like RAG presents challenges related to knowledge revision, interpretability, and avoiding hallucinations. Consider the difficulty in ensuring that a language model combines its pre-trained knowledge with dynamically retrieved information, avoiding inaccuracies and maintaining coherence.
- Knowledge Update Mechanisms:** Developing mechanisms to update non-parametric memory as real-world knowledge evolves is crucial. Imagine a scenario where RAG models need to adapt to changing information in domains like medicine, where new research findings and treatments continually emerge, requiring timely updates for accurate responses.

Improving RAG components (Ingestion)



1. Better Chunking Strategies



In the context of enhancing the Ingestion process for the RAG components, adopting advanced chunking strategies is necessary for efficient handling of textual data. In a simple RAG pipeline, a fixed strategy is adopted, i.e., a fixed number of words or characters form a single chunk.

Considering the complexities involved in large datasets, the following strategies are being used recently:

- Content-Based Chunking:** Breaks down text based on meaning and sentence structure using techniques like part-of-speech tagging or syntactic parsing. This preserves the sense and coherence of the text. However, one consideration of this chunking is it requires additional computational resources and algorithmic complexity.
- Sentence Chunking:** Involves breaking text into complete and grammatically correct sentences using sentence boundary recognition or speech segment. Maintains the unity and completeness of the text but can generate chunks of varying sizes, lacking homogeneity.

3. **Recursive Chunking:** Splits text into chunks of different levels, creating a hierarchical and flexible structure. Offers greater granularity and variety in text, but managing and indexing these chunks involves increased complexity.

2. Better Indexing Strategies



Improved indexing allows for more efficient search and retrieval of information. When chunks of data are properly indexed, it becomes easier to locate and retrieve specific pieces of information quickly. Some improved strategies include:

1. **Detailed Indexing:** Chunks through sub-parts (e.g., sentences) and assigns each chunk an identifier based on its position and a feature vector based on content. Provides specific context and accuracy but requires more memory and processing time.
2. **Question-Based Indexing:** Chunks through knowledge domains (e.g., topics) and assigns each chunk an identifier based on its category and a vector of characteristics based on relevance. Aligns directly with user requests, enhancing efficiency, but may result in information loss and lower accuracy.
3. **Optimized Indexing with Chunk Summaries:** Generates a summary for each chunk using extraction or compression techniques. Assigns an identifier based on the summary and a feature vector based on similarity. Provides greater synthesis and variety but demands complexity in generating and comparing summaries.

Improving RAG components (Retrieval)



1. Hypothetical Questions and HyDE:



The introduction of hypothetical questions involves generating a question for each chunk, embedding these questions in vectors, and performing a query search against this index of question vectors. This enhances search quality due to higher semantic similarity between queries and hypothetical questions compared to actual chunks. Conversely, HyDE (Hypothetical Response Extraction) involves generating a hypothetical response given the query, enhancing search quality by leveraging the vector representation of the query and its hypothetical response.

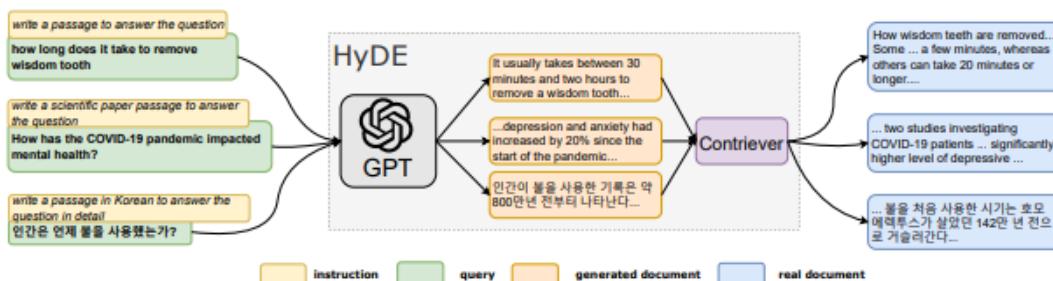


Image Source: https://arxiv.org/pdf/2212.10496.pdf

2. Context Enrichment:



The strategy here aims for smaller chunk retrieval for improved search quality while incorporating surrounding context for reasoning by the Language Model. Two options can be explored:

1. Sentence Window Retrieval: Embedding each sentence in a document separately to achieve high accuracy in the cosine distance search between the query and the context. After retrieving the most relevant single sentence, a context window is extended by including a specified number of sentences before and after the retrieved sentence. This extended context is then sent to the LLM for reasoning upon the provided query. The goal is to enhance the LLM's understanding of the context surrounding the retrieved sentence, enabling more informed responses.

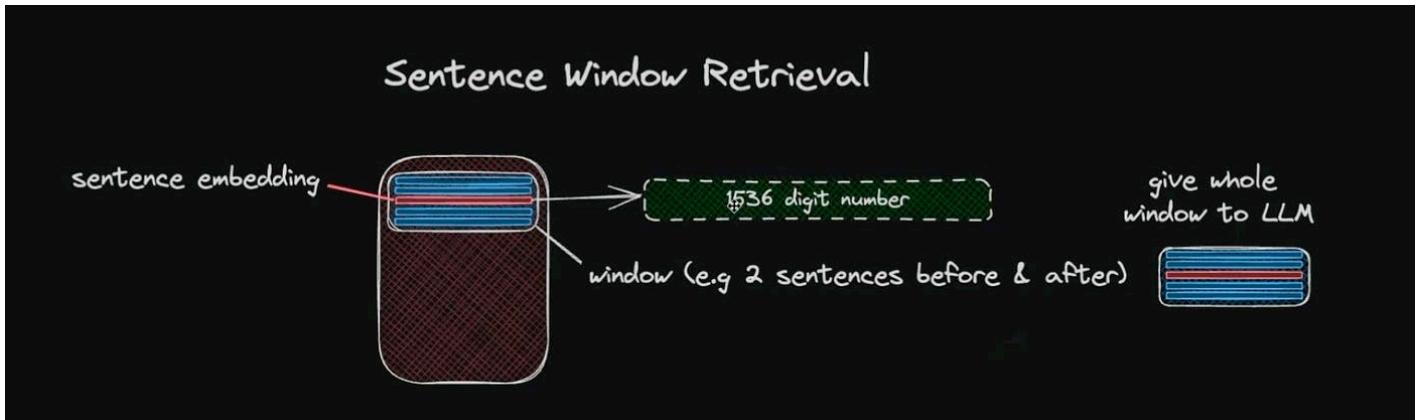


Image Source: <https://medium.com/@shivansh.kaushik/advanced-text-retrieval-with-elasticsearch-llamaindex-sentence-window-retrieval-cb5ea720aa44>

1. Auto-Merging Retriever: In this approach, documents are initially split into smaller child chunks, each referring to a larger parent chunk. During retrieval, smaller chunks are fetched first. If, among the top retrieved chunks, more than a specified number are linked to the same parent node (larger chunk), the context fed to the LLM is replaced by this parent node. This process can be likened to automatically merging several retrieved chunks into a larger parent chunk, hence the name "auto-merging retriever." The method aims to capture both granularity and context, contributing to more comprehensive and coherent responses from the LLM.

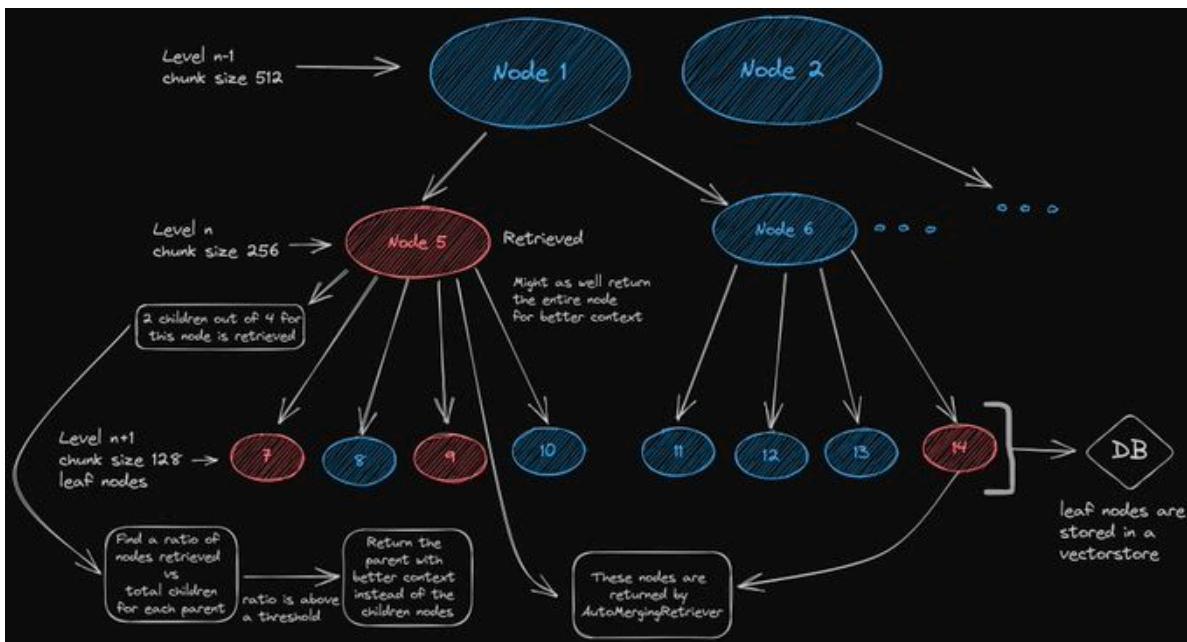


Image Source: <https://twitter.com/clusteredbytes>

3. Fusion Retrieval or Hybrid Search:



This strategy integrates conventional keyword-based search approaches with contemporary semantic search techniques. By incorporating diverse algorithms like tf-idf (term frequency-inverse document frequency) or BM25 alongside vector-based search, RAG systems can harness the benefits of both semantic relevance and keyword matching, resulting in more thorough and inclusive search outcomes.

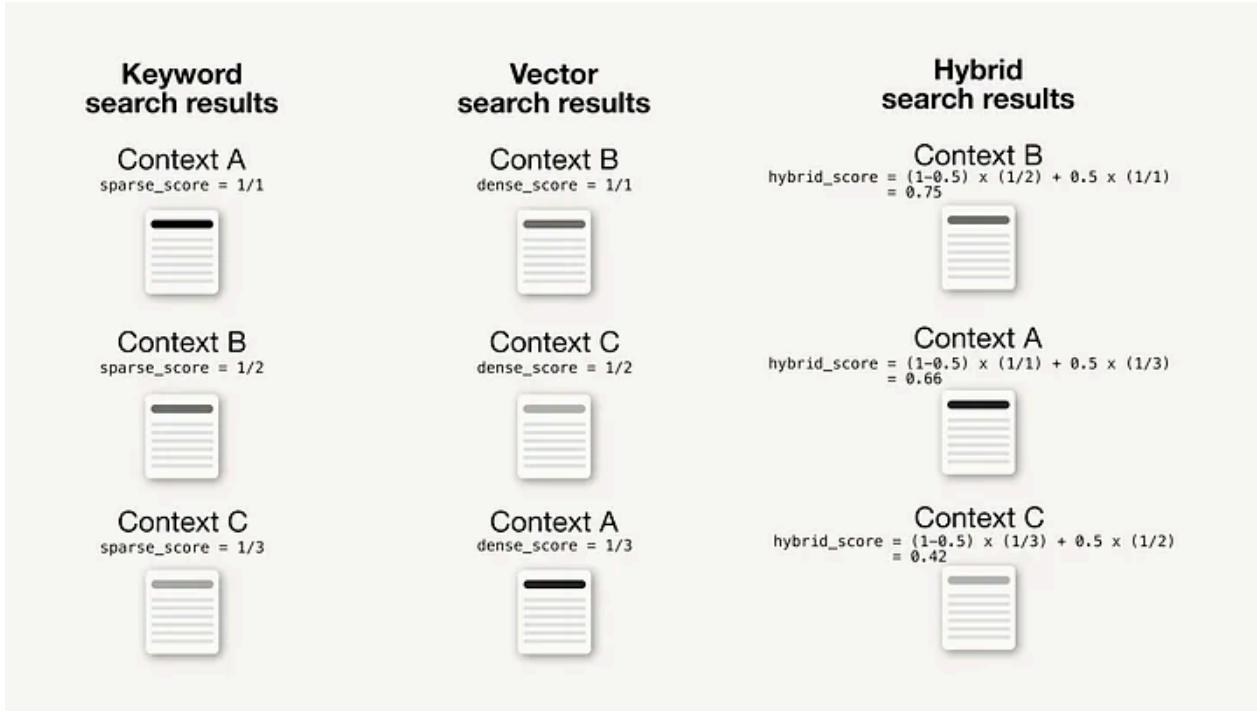


Image Source: <https://towardsdatascience.com/improving-retrieval-performance-in-rag-pipelines-with-hybrid-search-c75203c2f2f5>

4. Reranking & Filtering:



Post-retrieval refinement is performed through filtering, reranking, or transformations. LlamaIndex provides various Postprocessors, allowing the filtering of results based on similarity score, keywords, metadata, or reranking with models like LLMs or sentence-transformer cross-encoders. This step precedes the final presentation of retrieved context to the LLM for answer generation.

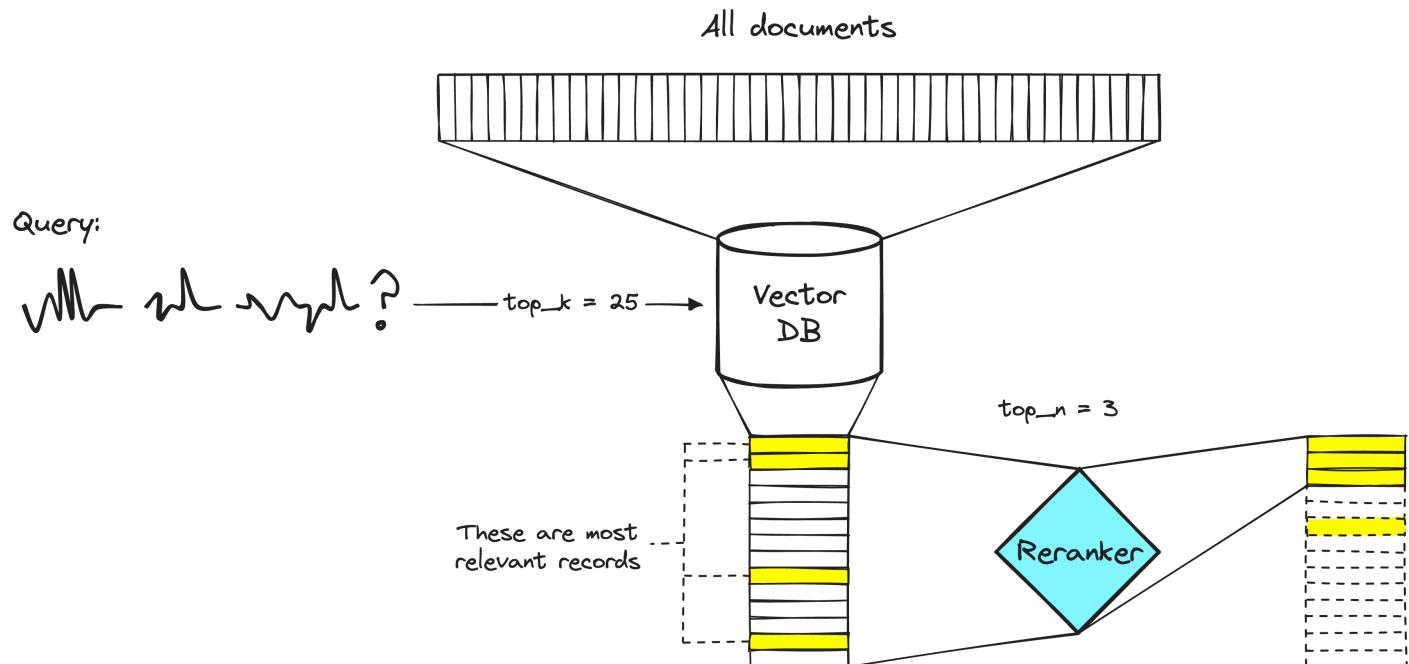


Image Source: <https://www.pinecone.io/learn/series/rag/rerankers/>

4. Query Transformations and Routing [Source]



Query transformation methods enhance retrieval by breaking down complex queries into sub-questions (Expansion) and improving poorly worded queries through re-writing. While dynamic Query Routing optimizes data retrieval in diverse

sources. The below are popular approaches

Query Transformations



1. **Query Expansion**: Query expansion decomposes the input into sub-questions, each of which is a more narrow retrieval challenge. For example, a question about physics can be stepped-back into a question (and LLM-generated answer) about the physical principles behind the user query.
2. **Query Re-writing**: Addressing poorly framed or worded user queries, the [Rewrite-Retrieve-Read](#) approach involves rephrasing questions to enhance retrieval effectiveness. The method is explained in detail in the paper.
3. **Query Compression**: In scenarios where a user question follows a broader chat conversation, the full conversational context may be necessary to answer the question. Query compression is utilized to condense chat history into a final question for retrieval.

Query Routing



1. **Dynamic Query Routing**: The question of where the data resides is crucial in RAG, especially in production settings with diverse data-stores. Dynamic query routing, supported by LLMs, efficiently directs incoming queries to the appropriate datastores. This dynamic routing adapts to different sources and optimizes the retrieval process.

Improving RAG components (Generation)



The most straightforward method for LLM generation involves concatenating all the relevant context pieces, surpassing a predefined relevance threshold, and presenting them along with the query to the LLM in a single instance. However, more advanced alternatives exist, necessitating multiple calls to the LLM to iteratively enhance the retrieved context, ultimately leading to the generation of a more refined and improved answer. Some methods are illustrated below.

1. Response Synthesis Approaches:



Involves 3 steps

1. **Iterative Refinement**: Refine the answer by sending retrieved context to the Language Model chunk by chunk.
2. **Summarization**: Summarize the retrieved context to fit into the prompt and generate a concise answer.
3. **Multiple Answers and Concatenation**: Generate multiple answers based on different context chunks and then concatenate or summarize them.

2. Encoder and LLM Fine-Tuning:



This approach involves the fine-tuning the LLM models within our RAG pipeline.

1. **Encoder Fine-Tuning**: Fine-tune the Transformer Encoder for better embeddings quality and context retrieval.
2. **Ranker Fine-Tuning**: Use a cross-encoder for reranking retrieved results, especially if there's a lack of trust in the base Encoder.
3. **RA-DIT Technique**: Use a technique like RA-DIT to tune both the LLM and the Retriever on triplets of query, context, and answer.

Read/Watch These Resources (Optional)



1. Building Production Ready RAG Applications: <https://www.youtube.com/watch?v=TRjq7t2Ms5I>
2. Amazon article on RAG- <https://docs.aws.amazon.com/sagemaker/latest/dg/jumpstart-foundation-models-customize-rag.html>
3. Huggingface tools for RAG- https://huggingface.co/docs/transformers/model_doc/rag

4. 12 RAG Pain Points and Proposed Solutions- <https://towardsdatascience.com/12-rag-pain-points-and-proposed-solutions-43709939a28c>

Read These Papers (Optional)



1. [Retrieval-Augmented Generation for Large Language Models: A Survey](#)
2. [Seven Failure Points When Engineering a Retrieval Augmented Generation System](#)

CSE 561 Paper Review

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis^{†‡}, Ethan Perez^{*},

Aleksandra Piktus[†], Fabio Petroni[†], Vladimir Karpukhin[†], Naman Goyal[†], Heinrich Küttler[†],

Mike Lewis[†], Wen-tau Yih[†], Tim Rocktäschel^{†‡}, Sebastian Riedel^{†‡}, Douwe Kiela[†]

[†]Facebook AI Research; [‡]University College London; ^{*}New York University;

plewis@fb.com

GENERALIZATION THROUGH MEMORIZATION: NEAREST NEIGHBOR LANGUAGE MODELS

Urvashi Khandelwal[†], Omer Levy[‡], Dan Jurafsky[†], Luke Zettlemoyer[‡] & Mike Lewis[‡]

[†]Stanford University

[‡]Facebook AI Research

{urvashik, jurafsky}@stanford.edu

{omerlevy, lsz, mikelewis}@fb.com

Overview

- Retrieval Augmented Generation (RAG)
- Model Setup
- Results
- Performance comparison

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis^{†‡}, Ethan Perez^{*},

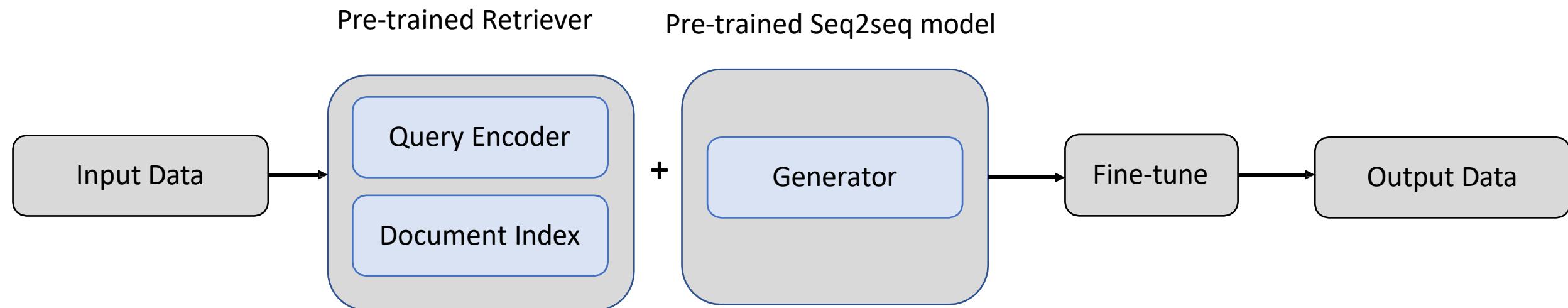
Aleksandra Piktus[†], Fabio Petroni[†], Vladimir Karpukhin[†], Naman Goyal[†], Heinrich Küttler[†],

Mike Lewis[†], Wen-tau Yih[†], Tim Rocktäschel^{†‡}, Sebastian Riedel^{†‡}, Douwe Kiela[†]

[†]Facebook AI Research; [‡]University College London; ^{*}New York University;
plewis@fb.com

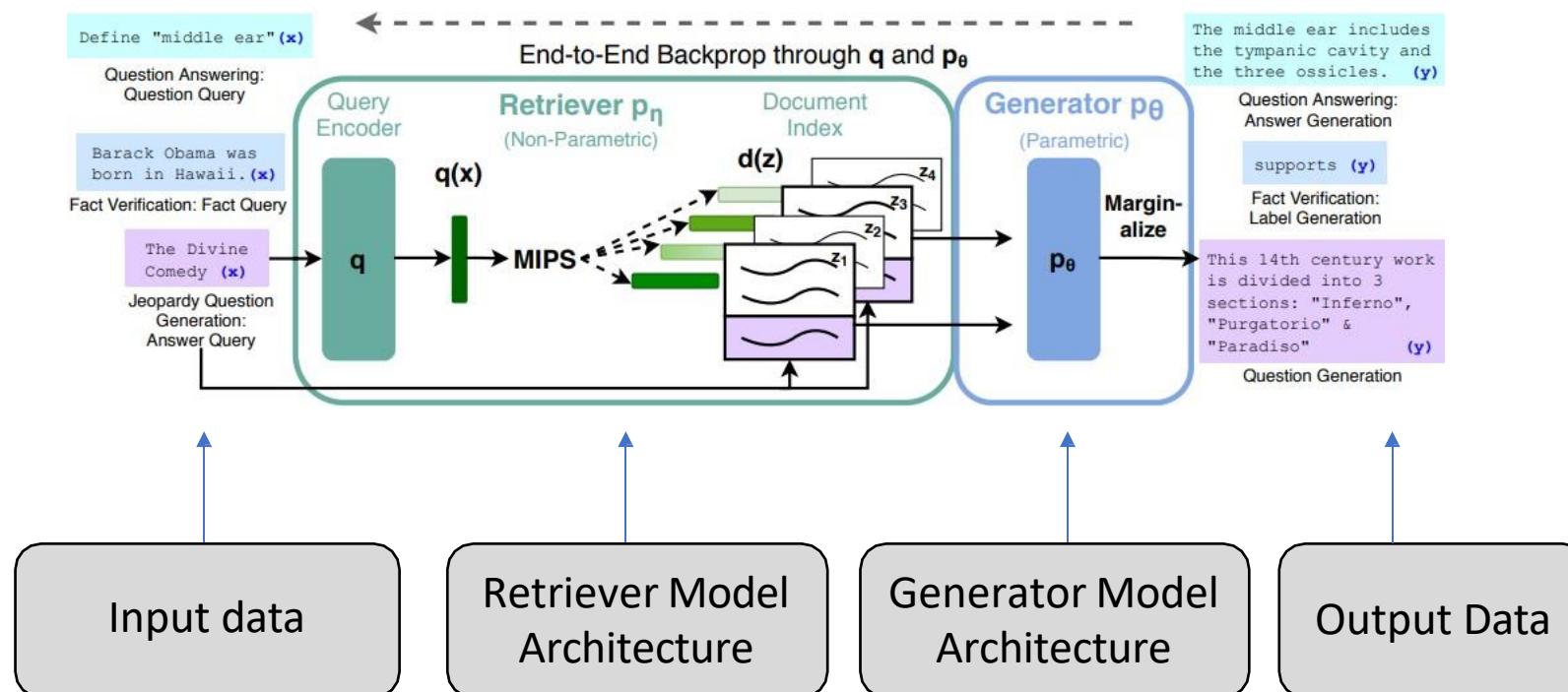
Retrieval Augmented Generation (RAG)

- Retrieval-augmented generation (RAG) is a technique for enhancing the accuracy and reliability of generative AI models with facts fetched from external sources



Importance

- Updated world knowledge
- Providing insights into their predictions
- Hallucinations



For query x , Maximum Inner Product Search (MIPS) is used to find the **top-K** documents z_i .

For final prediction y , z is treated as a latent variable and marginalized over seq2seq predictions given different documents

RAG implementations showed better performance in all tested tasks .

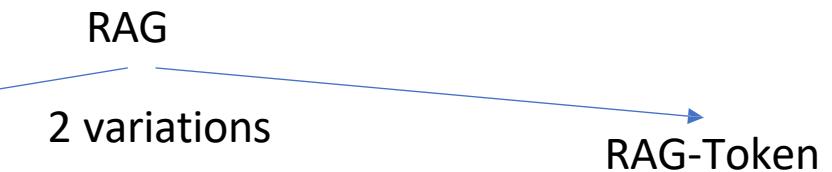
Models

RAG-Sequence
Retrieved passages are treated as sequences of text

Each retrieved passage is concatenated with the input data or prompt to form a longer sequence

This combined sequence is then fed into the generator model (e.g., BART) for generating the final output

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y|x, z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) \prod_i^N p_\theta(y_i|x, z, y_{1:i-1})$$



Retrieved passages are represented as token-level embeddings

Instead of concatenating the passages with the input data, token-level representations of the passages are directly integrated into the input embeddings

The generator model (BART) then operates on these modified input embeddings, considering the additional information from the retrieved passages during generation.

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y_i|x, z_i, y_{1:i-1})$$

Experimental Results

- Tasks:
 - Open-domain Question Answering
 - Abstractive Question Answering
 - Jeopardy Question Generation
 - Fact Verification

Experimental Results

Task: Open-domain Question Answering

	Model	NQ	TQA	WQ	CT
Closed	T5-11B [52]	34.5	- /50.1	37.4	-
Book	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open	REALM [20]	40.4	- / -	40.7	46.8
Book	DPR [26]	41.5	57.9 / -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	45.5	50.0
	RAG-Seq.	44.5	56.8/ 68.0	45.2	52.2

Table 1: Open-Domain QA Test Scores.

Model	Jeopardy		MSMARCO		FVR3	FVR2
	B-1	QB-1	R-L	B-1	Label Acc.	
SotA	-	-	49.8*	49.9*	76.8	92.2*
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok.	17.3	22.2	40.1	41.5	72.5	<u>89.5</u>
RAG-Seq.	14.7	21.4	<u>40.8</u>	<u>44.2</u>		

Table 2: Generation and classification Test Scores.

RAG implementations showed better performance showing improved performance on all except one open-domain question answering task.

Experimental Results

Task: Fact Verification

Document 1: his works are considered classics of American literature ... His wartime experiences formed the basis for his novel "A Farewell to Arms" (1929) ...

Document 2: ... artists of the 1920s "Lost Generation" expatriate community. His debut novel, **"The Sun Also Rises"**, was published in 1926.

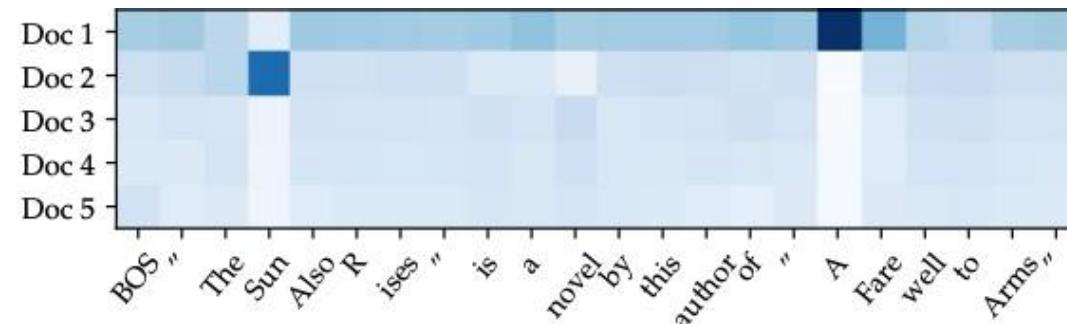


Figure 2: RAG-Token document posterior $p(z_i|x, y_i, y_i)$ for each generated token for input "Hemingway" for Jeopardy generation with 5 retrieved documents.

Experimental Results

Task: Fact Verification

Task	Input	Model	Generation
MS-MARCO	define middle ear	BART	?The middle ear is the part of the ear between the middle ear and the nose.
		RAG-T	The middle ear is the portion of the ear internal to the eardrum.
		RAG-S	The middle ear includes the tympanic cavity and the three ossicles.
Jeopardy Question Generation	what currency needed in scotland	BART	The currency needed in Scotland is Pound sterling.
		RAG-T	Pound is the currency needed in Scotland.
		RAG-S	The currency needed in Scotland is the pound sterling.
Jeopardy Question Generation	Washington	BART	?This state has the largest number of counties in the U.S.
		RAG-T	It's the only U.S. state named for a U.S. president
		RAG-S	It's the state where you'll find Mount Rainier National Park
Jeopardy Question Generation	The Divine Comedy	BART	*This epic poem by Dante is divided into 3 parts: the Inferno, the Purgatorio & the Purgatorio
		RAG-T	Dante's "Inferno" is the first part of this epic poem
		RAG-S	This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso"

Table 3: Examples from generation tasks. RAG models generate more specific and factually accurate responses. '?' indicates factually incorrect responses, '*' indicates partially correct responses.

Experimental Results

Task: Jeopardy Question Generation

	Factuality	Specificity
BART better	7.1%	16.8%
RAG better	42.7%	37.4%
Both good	11.7%	11.8%
Both poor	17.7%	6.9%
No majority	20.8%	20.1%

Table 4: Human assessments for the Jeopardy Question Generation Task

Task: Generation Diversity

	MSMARCO	Jeopardy QGen
Gold	89.6%	90.0%
BART	70.7%	32.4%
RAG-Token	77.8%	46.8%
RAG-Seq.	83.5%	53.8%

Table 5: Ratio of distinct to total tri-grams for generation tasks

Experimental Results

Task: Retrieval Ablations

Model	NQ	TQA Exact Match	WQ	CT	Jeopardy-QGen		MSMarco		FVR-3 Label Accuracy	FVR-2
					B-1	QB-1	R-L	B-1		
RAG-Token-BM25	29.7	41.5	32.1	33.1	17.5	22.3	55.5	48.4		
RAG-Sequence-BM25	31.8	44.1	36.6	33.8	11.1	19.5	56.5	46.9	75.1	91.6
RAG-Token-Frozen	37.8	50.1	37.1	51.1	16.7	21.7	55.9	49.4		
RAG-Sequence-Frozen	41.2	52.1	41.8	52.6	11.8	19.6	56.7	47.3	72.9	89.4
RAG-Token	43.5	54.8	46.5	51.9	17.9	22.6	56.2	49.4		
RAG-Sequence	44.0	55.8	44.9	53.4	15.3	21.5	57.2	47.5	74.5	90.6

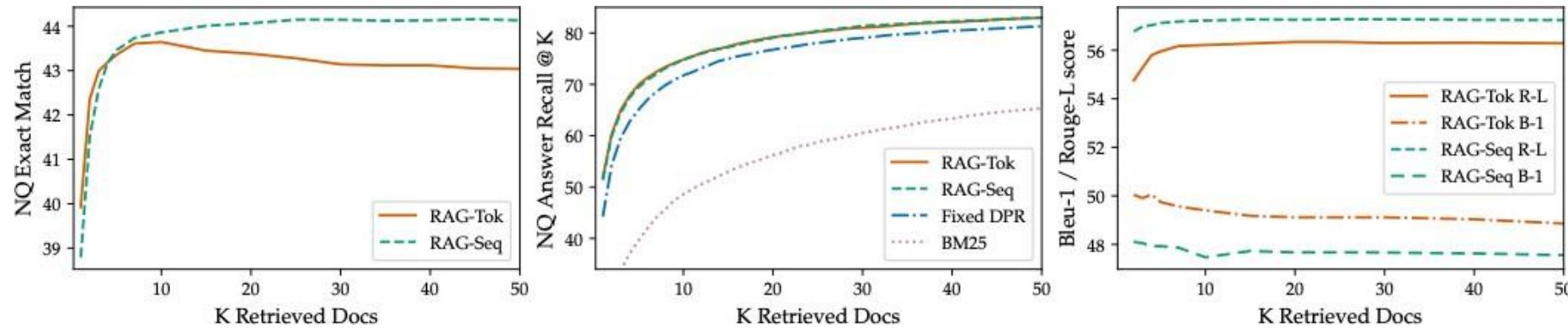
Table 6: Ablations on the dev set. As FEVER is a classification task, both RAG models are equivalent.

Experimental Results

Task: Index hot-swapping

- Built an index using the DrQA Wikipedia dump from December 2016 and compare outputs from RAG using this index to the newer index from our main results (December 2018).
- RAG answers 70% correctly using the 2016 index for 2016 world leaders and 68% using the 2018 index for 2018 world leaders.
- This shows that RAG's world knowledge can be updated by simply replacing its non-parametric memory.

Effect of Retrieving more documents



Retrieving more documents can lead to improved relevance of the retrieved passages. The model shows diminishing returns when it comes to the number of documents retrieved after retrieving a certain number of documents.

Conclusion

- Hybrid generation models with access to parametric and non-parametric memory.
- Obtains state of the art results on open-domain QA
- Improved generation compared to parametric BART, with RAG more factual and specific

Overview

- kNN-LM
- Model Setup
- Experimental Results
- Performance comparison

GENERALIZATION THROUGH MEMORIZATION: NEAREST NEIGHBOR LANGUAGE MODELS

Urvashi Khandelwal^{†,*}, Omer Levy[‡], Dan Jurafsky[†], Luke Zettlemoyer[‡] & Mike Lewis[‡]

[†]Stanford University

[‡]Facebook AI Research

{urvashik, jurafsky}@stanford.edu

{omerlevy, lsz, mikelewis}@fb.com

kNN-LM

- An approach that extends a pre-trained LM by linearly interpolating its next word distribution with a k-nearest neighbors (kNN) model
- The nearest neighbors are computed according to distance in the pre-trained embedding space and can be drawn from any text collection, including the original LM training data.
- This approach allows rare patterns to be memorized explicitly, rather than implicitly in model parameters.
- It also improves performance when the same training data is used for learning the prefix representations and the kNN model, strongly suggesting that the prediction problem is more challenging than previously appreciated.

Model Architecture

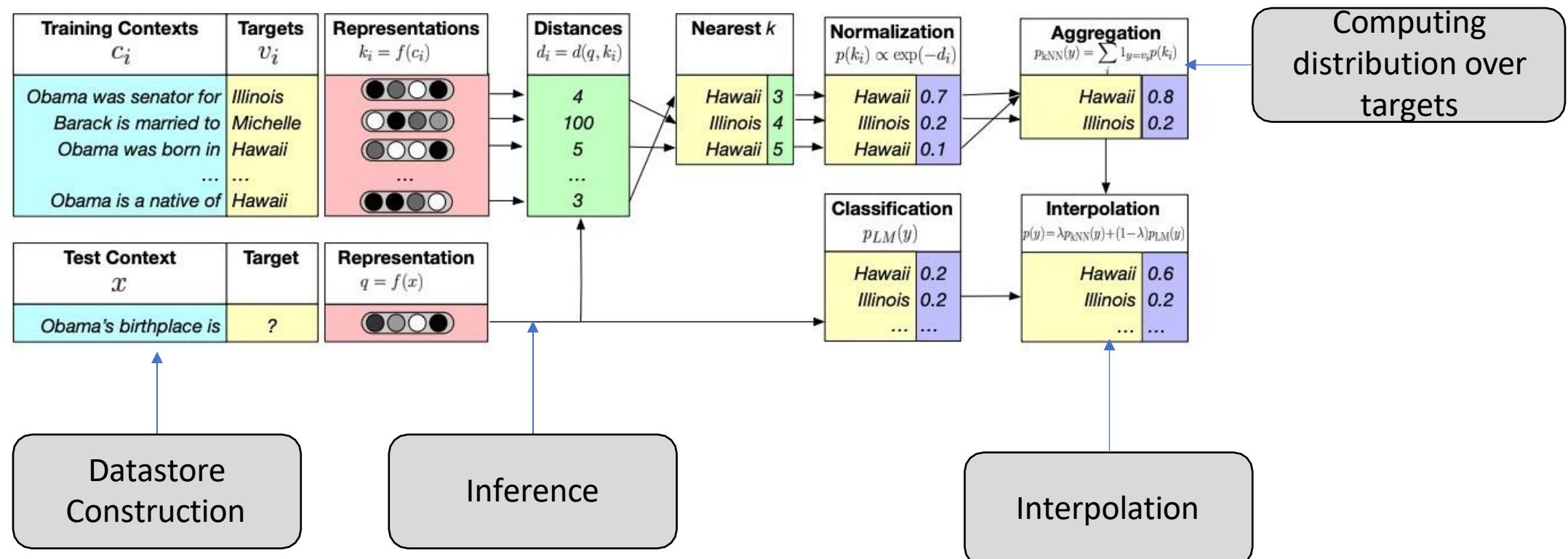


Figure 1. Illustration of kNN-LM

Experimental Results

Model	Perplexity (↓)		# Trainable Params
	Dev	Test	
Baevski & Auli (2019)	17.96	18.65	247M
+Transformer-XL (Dai et al., 2019)	-	18.30	257M
+Phrase Induction (Luo et al., 2019)	-	17.40	257M
Base LM (Baevski & Auli, 2019)	17.96	18.65	247M
+kNN-LM	16.06	16.12	247M
+Continuous Cache (Grave et al., 2017c)	17.67	18.27	247M
+kNN-LM + Continuous Cache	15.81	15.79	247M

Table 1. Performance on WIKITEXT-103

Model	Perplexity (↓)		# Trainable Params
	Dev	Test	
Base LM (Baevski & Auli, 2019)	14.75	11.89	247M
+kNN-LM	14.20	10.89	247M

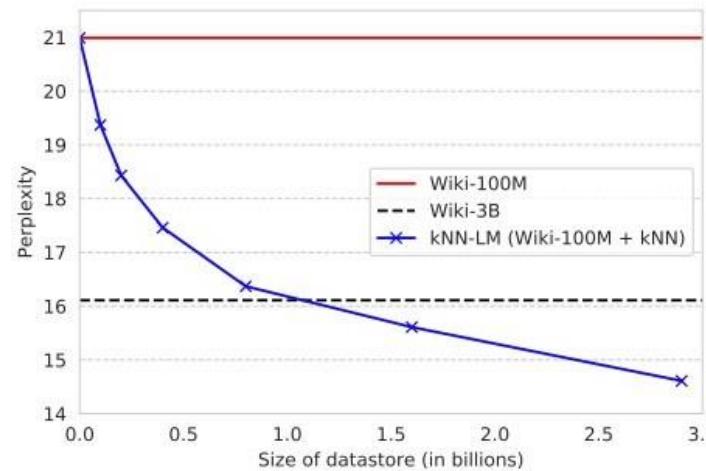
Table 2. Performance on BOOKS

The kNN-LM model shows improvement compare to the baselines with lower perplexity scores

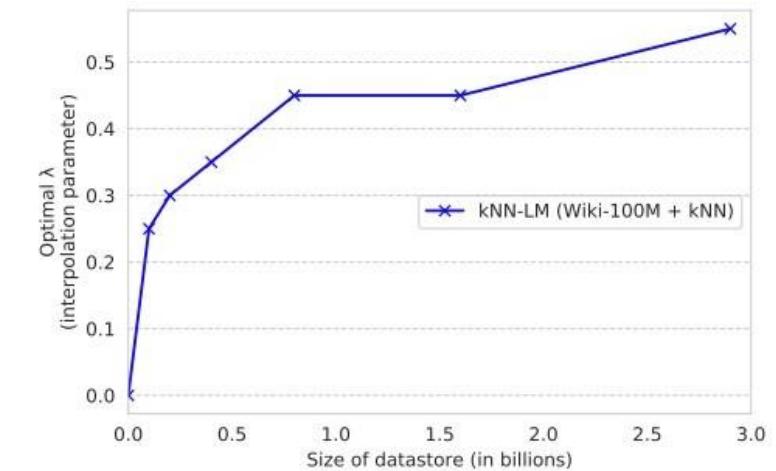
Experimental Results

Task : Training Data as the datastore

Training Data	Datastore	Perplexity (\downarrow)	
		Dev	Test
WIKI-3B	-	16.11	15.17
WIKI-100M	-	20.99	19.59
WIKI-100M	WIKI-3B	14.61	13.73



(a) Effect of datastore size on perplexities.



(b) Tuned values of λ for different datastore sizes.

Table 3. Experimental results on WIKI-3B

Figure 2. Varying size on the datastore

As the size of the datastore increases, a higher weight on the retrieved training examples (controlled by λ) becomes more beneficial in improving model performance

Experimental Results

Task : Additional Data without training

Training Data	Datastore	Perplexity (↓)	
		Dev	Test
WIKI-3B	-	37.13	34.84
BOOKS	-	14.75	11.89
WIKI-3B	BOOKS	24.85	20.47

Table 4. Performance on in-domain BOOKS data

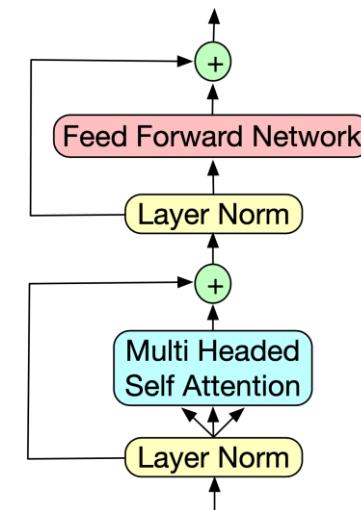


Figure 3. Transformer layer of the LM

Key Type	Dev ppl. (↓)
No datastore	17.96
Model output	17.07
Model output layer normalized	17.01
FFN input after layer norm	16.06
FFN input before layer norm	17.06
MHSA input after layer norm	16.76
MHSA input before layer norm	17.14

Table 5. WIKITEXT-103 validation results using different states from the final layer of the LM as the representation function for keys and queries

Experimental Results

Task: Tuning Nearest neighbor search

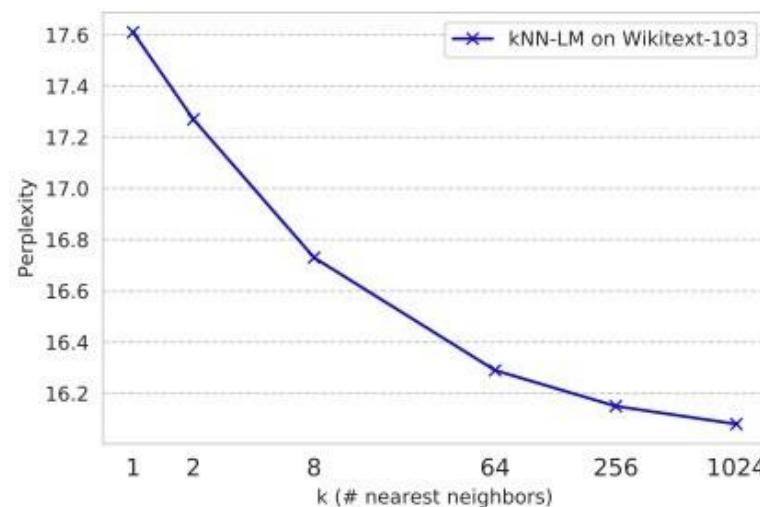


Figure 4. Effect of the number of nearest neighbors returned per word on WIKITEXT-103 (validation set).

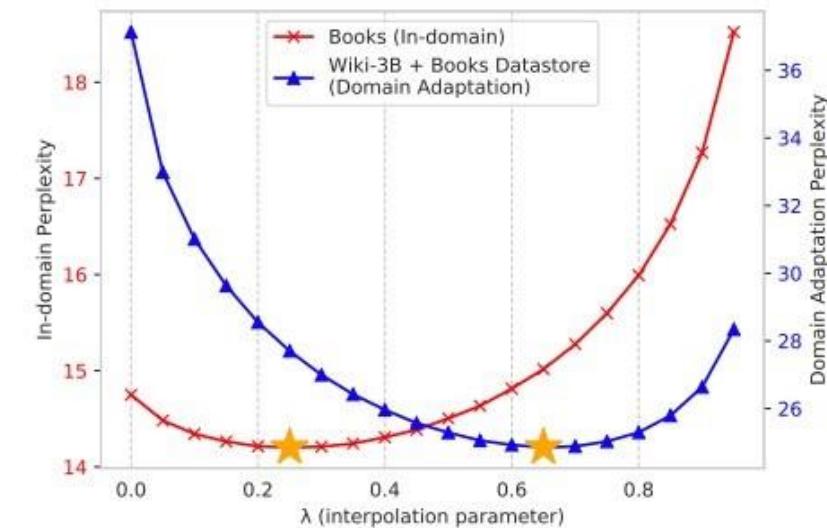


Figure 4. Effect of interpolation parameter λ on in-domain (left y-axis) and out-of-domain (right y-axis) validation set performances.

Experimental Results

Test Context ($p_{\text{kNN}} = 0.998, p_{\text{LM}} = 0.124$)	Test Target	
<i>it was organised by New Zealand international player Joseph Warbrick, promoted by civil servant Thomas Eyton, and managed by James Scott, a publican. The Natives were the first New Zealand team to perform a haka, and also the first to wear all black. They played 107 rugby matches during the tour, as well as a small number of Victorian Rules football and association football matches in Australia. Having made a significant impact on the...</i>	development	
Training Set Context	Training Set Target	Context Probability
<i>As the captain and instigator of the 1888-89 Natives – the first New Zealand team to tour the British Isles – Warbrick had a lasting impact on the...</i>	development	0.998
<i>promoted to a new first grade competition which started in 1900. Glebe immediately made a big impact on the...</i>	district	0.00012
<i>centuries, few were as large as other players managed. However, others contend that his impact on the...</i>	game	0.000034
<i>Nearly every game in the main series has either an anime or manga adaptation, or both. The series has had a significant impact on the...</i>	development	0.00000092

Figure 6: Example where the k NN model has much higher confidence in the correct target than the LM. Although there are other training set examples with similar local n -gram matches, the nearest neighbour search is highly confident of specific and very relevant context.

Figure 6. Example where the kNN model has much higher confidence in the correct target than the LM.

Experimental Results

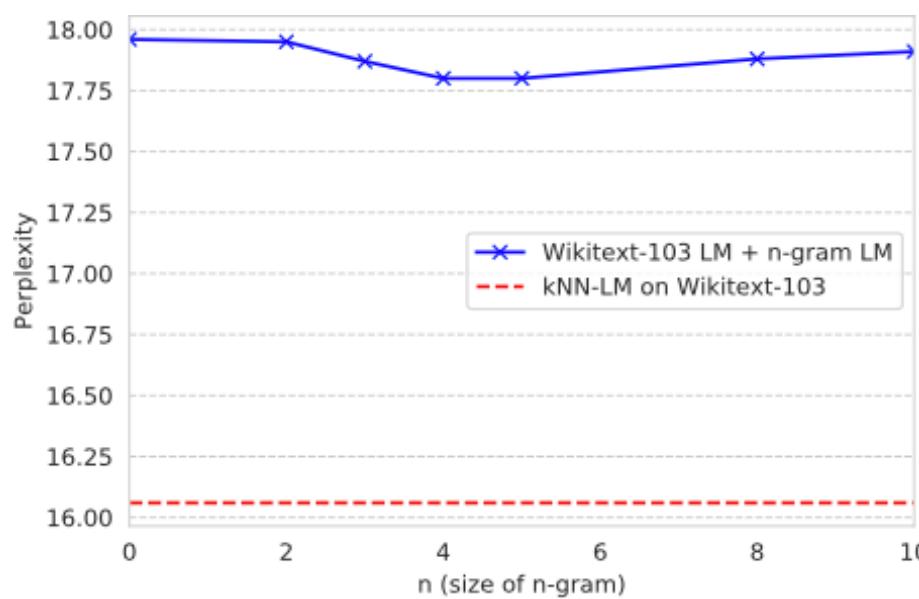


Figure 7. Interpolating the Transformer LM with n-gram LMs on WIKITEXT-103 (validation set).

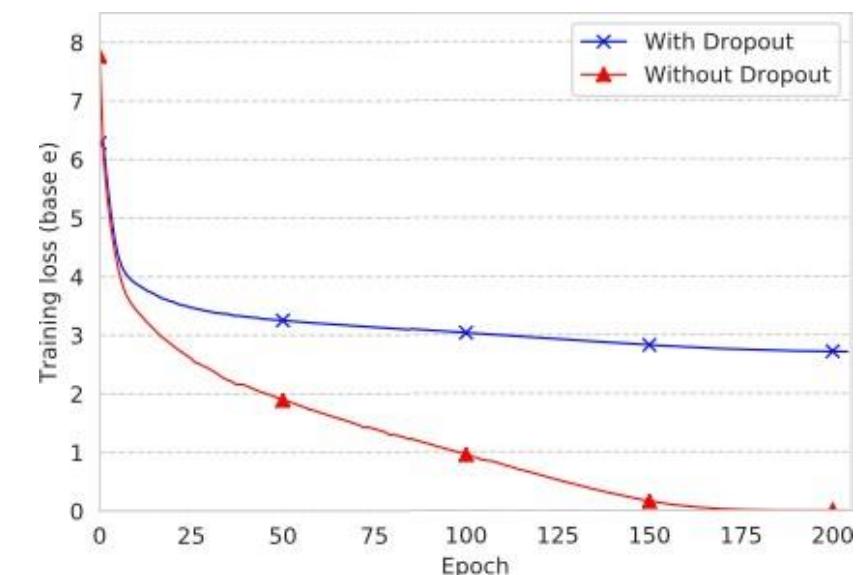


Figure 8. Training curves for the Transformer LM with and without dropout.

Conclusion

- k NN-LM outperform standard language models by directly querying training examples at test time
- Learning similarity functions between contexts may be an easier problem than predicting the next word from some given context.

Questions?

~~→ UX~~
→ Revision ✓
→ Doubts ✓

Mid - 19th

Conversational UX Design: Types and Examples

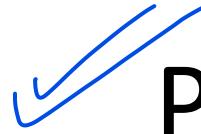
Improving User Experience through AI
and Technology

Date: Date

Presenter: Presenter

Introduction

- Conversational UX involves designing user experiences that incorporate conversational interfaces such as chatbots and voice assistants. It enhances user interaction by making technology more intuitive and responsive.



Principles of Conversational UX

- ✓ 1. Responsiveness: Quick and efficient responses.
- ✓ 2. Availability: 24/7 accessibility across devices.
- ✓ 3. Simplicity: Easy and intuitive interactions.
- ✓ 4. Exit Options: Providing users with a way to exit the conversation or reach human support.

Significance of Conversational UX

- User Benefits: Immediate issue resolution, multitasking capabilities.
- Organizational Benefits:
 - (1) Cost reduction, ✓
 - (2) Efficiency of existing tasks.
 - (3)

✓ Types of Conversational UX

- ✓ • 1. Chatbots: Automated text-based interactions.
- ✓ • 2. Interactive Applications: Mobile apps with interactive features.
- ✓ • 3. Digital Voice Assistants: AI-driven voice interactions (e.g., Siri, Alexa).

Example 1: Bank of America's Erica

- Erica is a virtual assistant for banking needs, improving customer service and task efficiency.

Example 2: Duolingo

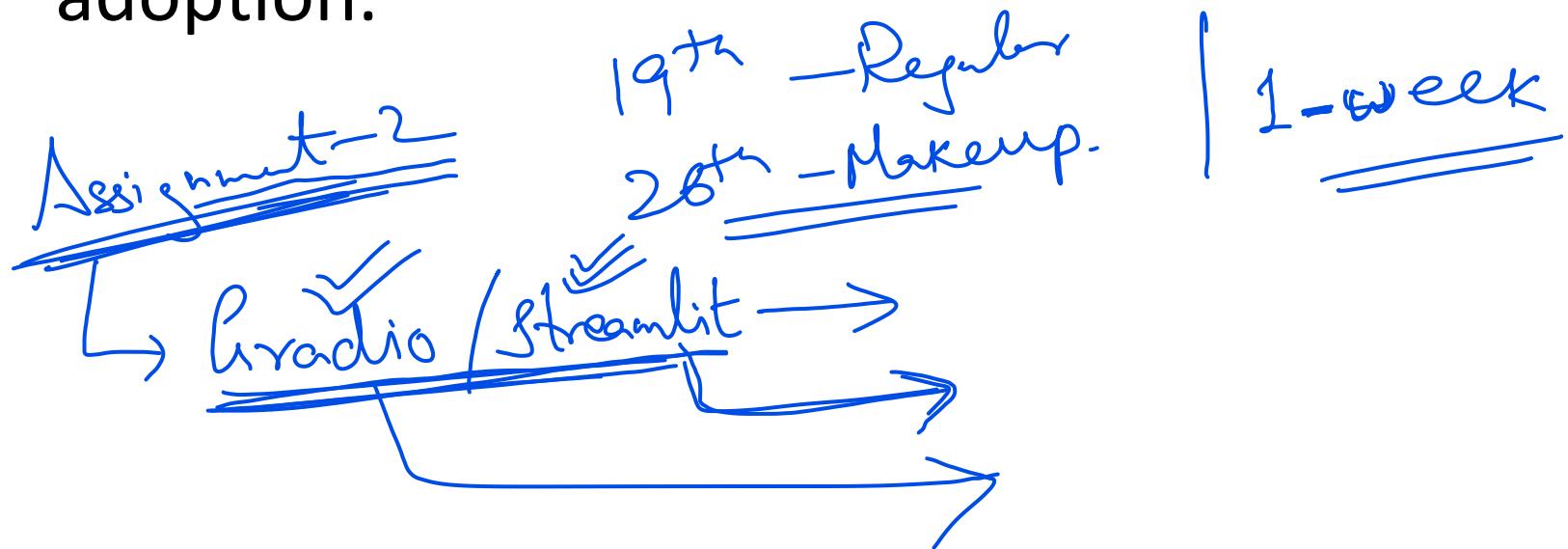
- Duolingo uses interactive exercises for language learning and has over 500 million downloads, making it a popular educational tool.

Example 3: Domino's Facebook Messenger Bot

- This chatbot allows ordering and tracking through Facebook Messenger, streamlining the ordering process.

Conclusion

- Conversational UX is a transformative tool for enhancing user interactions. The future holds potential for further advancements and wider adoption.



Q&A

- Questions & Answers
- Contact Information

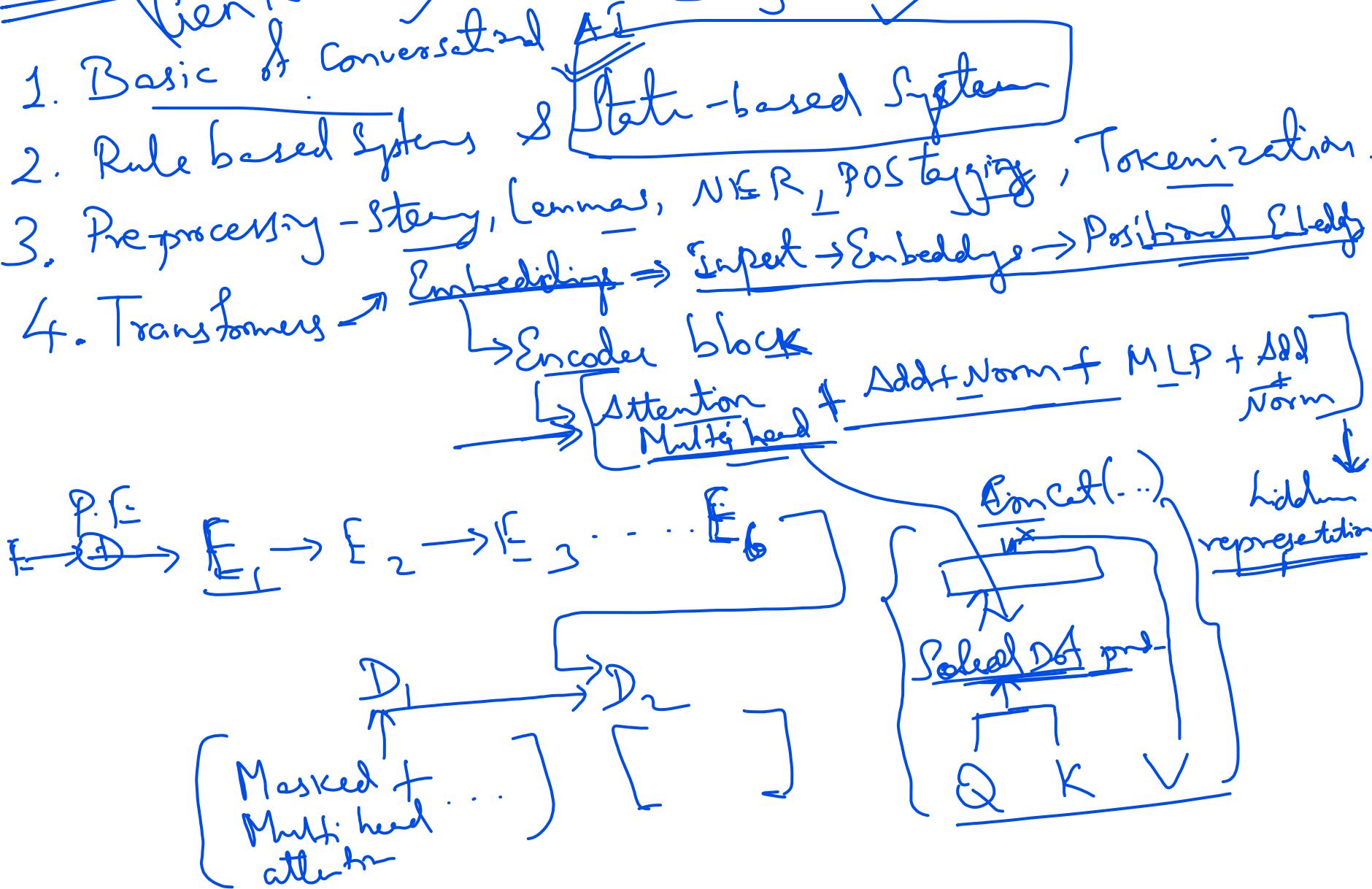
References

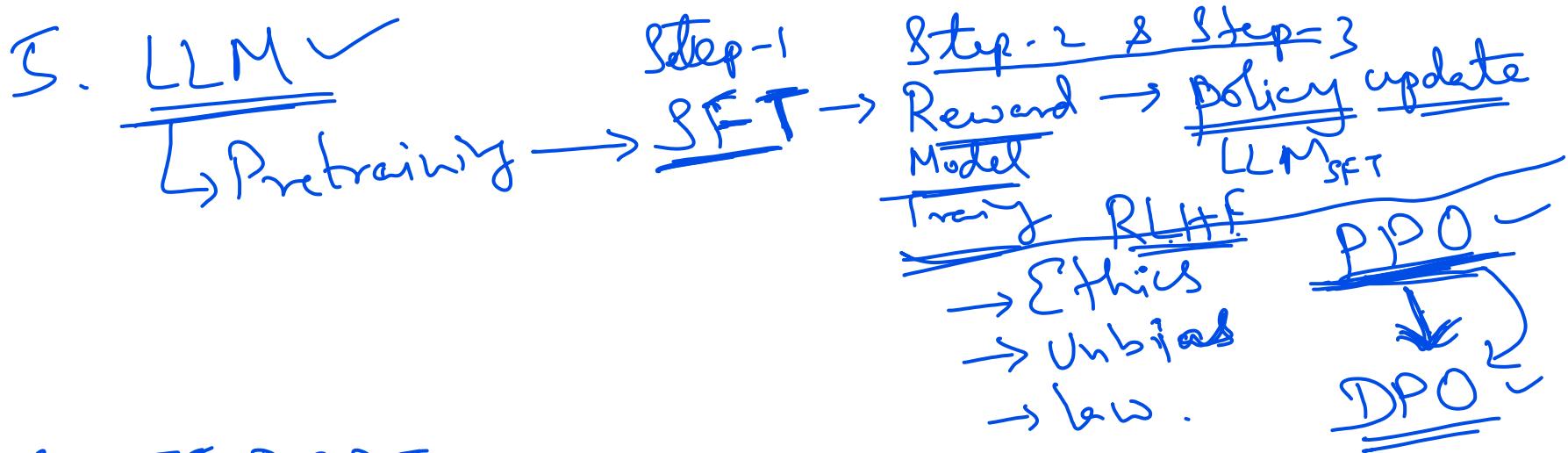
- Conversational UX Design: Types and Examples
- <https://www.ramotion.com/blog/conversational-ux-design/>

~~Revision:-
Chen AI~~

MCQ → 10 to 12 Marks.

Subjective → 18-20 marks.



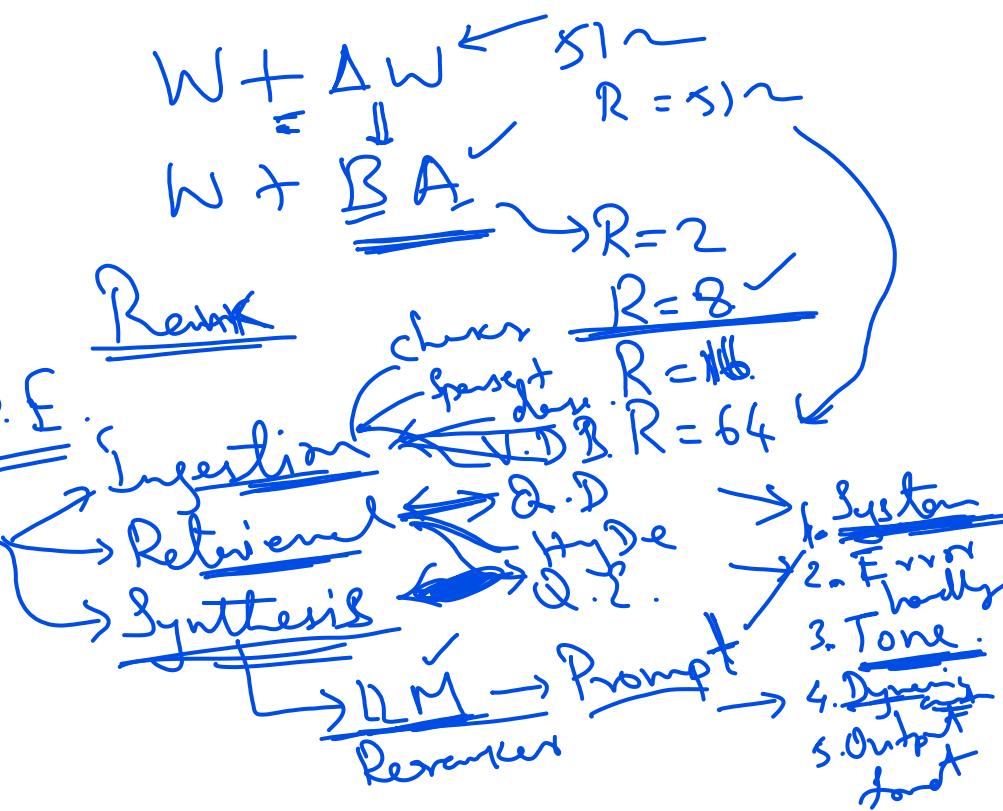


6. T5, BERT

7. Finetuning → LORA
(Adapters)

8. RAG - why RAG + P.E.
↳ Three phases

→ UX =



$4 \text{ Qs} \rightarrow R$

$\frac{1}{2} \rightarrow \text{MCQs} \rightarrow \text{Common, Slide}$

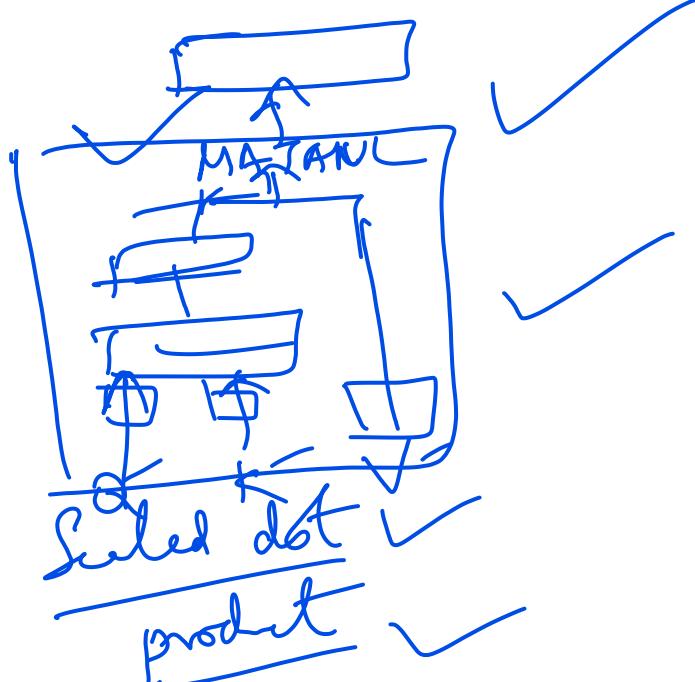
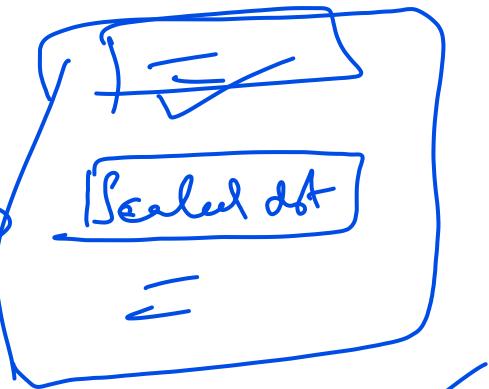
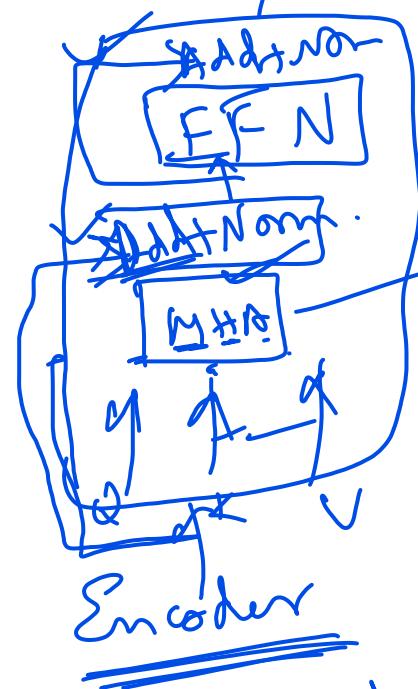
$\frac{1}{8} \rightarrow \text{Subjective}$

$4 \text{ Qs} \rightarrow M \text{ (bit)}$

$\frac{1}{16} \rightarrow 10 \text{ MCQs}$

$\frac{1}{32} \rightarrow 20 \text{ Subject}$

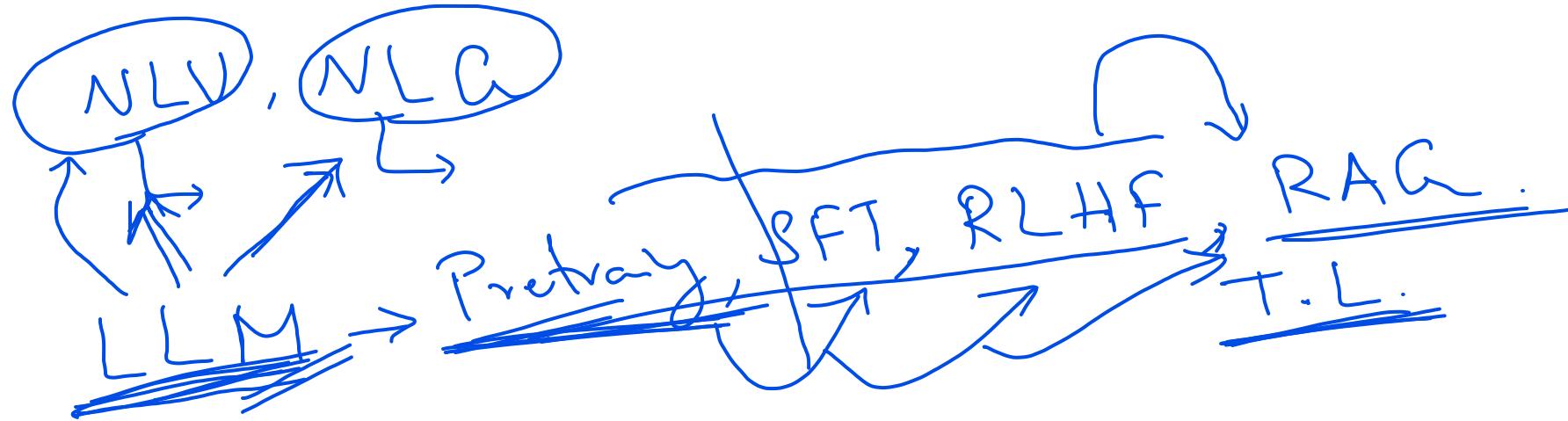
hidden Represent



Decoder

MMHA

Conv. AI
↳ 2 types



↳ Clinical
Regulations



1. Pretrain | fine-tune (RLHF) \Rightarrow Data \Rightarrow biased \Rightarrow LLM response \rightarrow
2. Model architecture \Rightarrow bias
3. Project / Assignment \Rightarrow Algo (cleaning, etc.) \Rightarrow Prompt .

Bias and UI/UX in LLMs

4.  votes \rightarrow Interaction with users (feed back) .

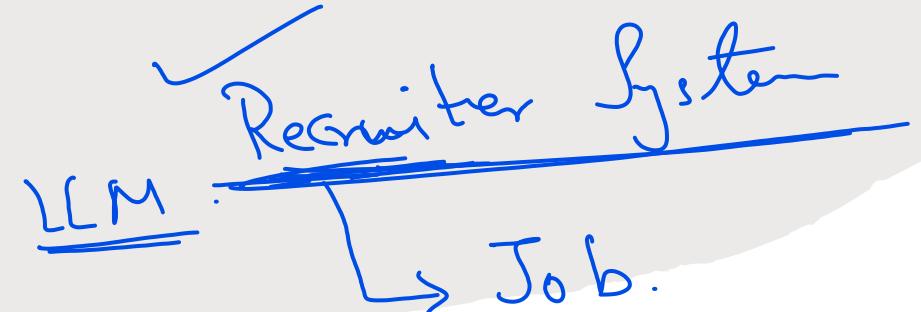
Bias

Bias in LLMs can stem from several sources:

- ✓ Training Data: LLMs are trained on vast amounts of text data from the internet, which inherently contains societal biases related to race, gender, culture, and religion.
- ✓ Model Specifications: The architecture and algorithms used can also introduce or amplify biases present in the training data.
- ✓ Algorithmic Constraints: Certain design choices and constraints in the model's development can lead to biased outcomes.
- ✓ Product Design and Policy Decisions: Decisions made during the product development phase, including how the model is fine-tuned and deployed, can influence the presence and impact of bias.

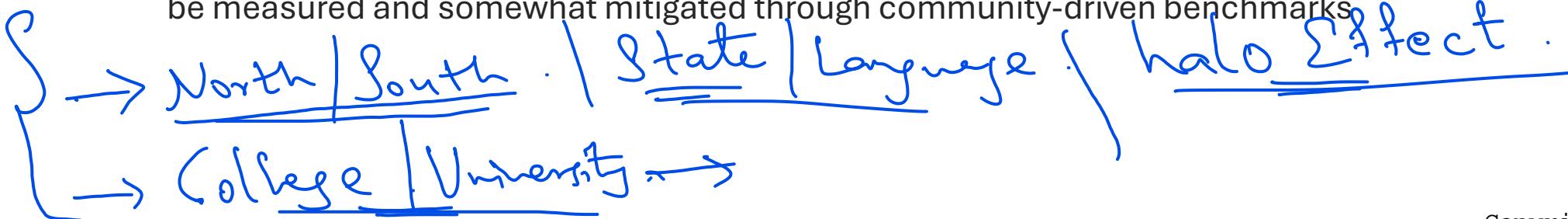
* Interaction with Users (after deployment) .
↳ Tuning

Types of Bias



Mitigate Bias in LLMs can manifest in various forms:

- Gender Bias: Associating certain professions or roles with specific genders. For example, tasks related to software development often show a gendered distribution, with technical roles being more frequently associated with men.
- Cultural Bias: Models may favor Western cultural norms and fail to appropriately adapt to other cultural contexts. For instance, multilingual models often exhibit biases towards Western culture over others.
- Religious Bias: LLMs can display biases against specific religious groups, such as associating Muslims with violence more frequently than other religious groups.
- LGBTQ+ Bias: Models may encode biases harmful to the LGBTQ+ community, which can be measured and somewhat mitigated through community-driven benchmarks.



Whose Opinions Do Language Models Reflect



language models have offered subjective opinions to controversial social and political queries



whose opinions (if any) do language models reflect?

How to measure Bias in LLMs?

- There are multiple metrics which can be used to measure bias in LLMs

- RBS and ABS
- BiQ

- For fairness the set of metrics used are:

- Demographic Parity
- Equality of odds
- Disparate Treatment
- Disparate Impact

Bias vs fairness.
Unbiased ≠ Fair.

1. Fair but biased → e.g.

2. Unbiased but unfair → e.g.

RBS and ABS

Representative Bias Score (RBS): This metric measures the extent to which LLMs generate outputs that reflect the experiences of certain identity groups over others. It helps in identifying if the model is biased towards specific demographics such as race, gender, or sexual orientation.

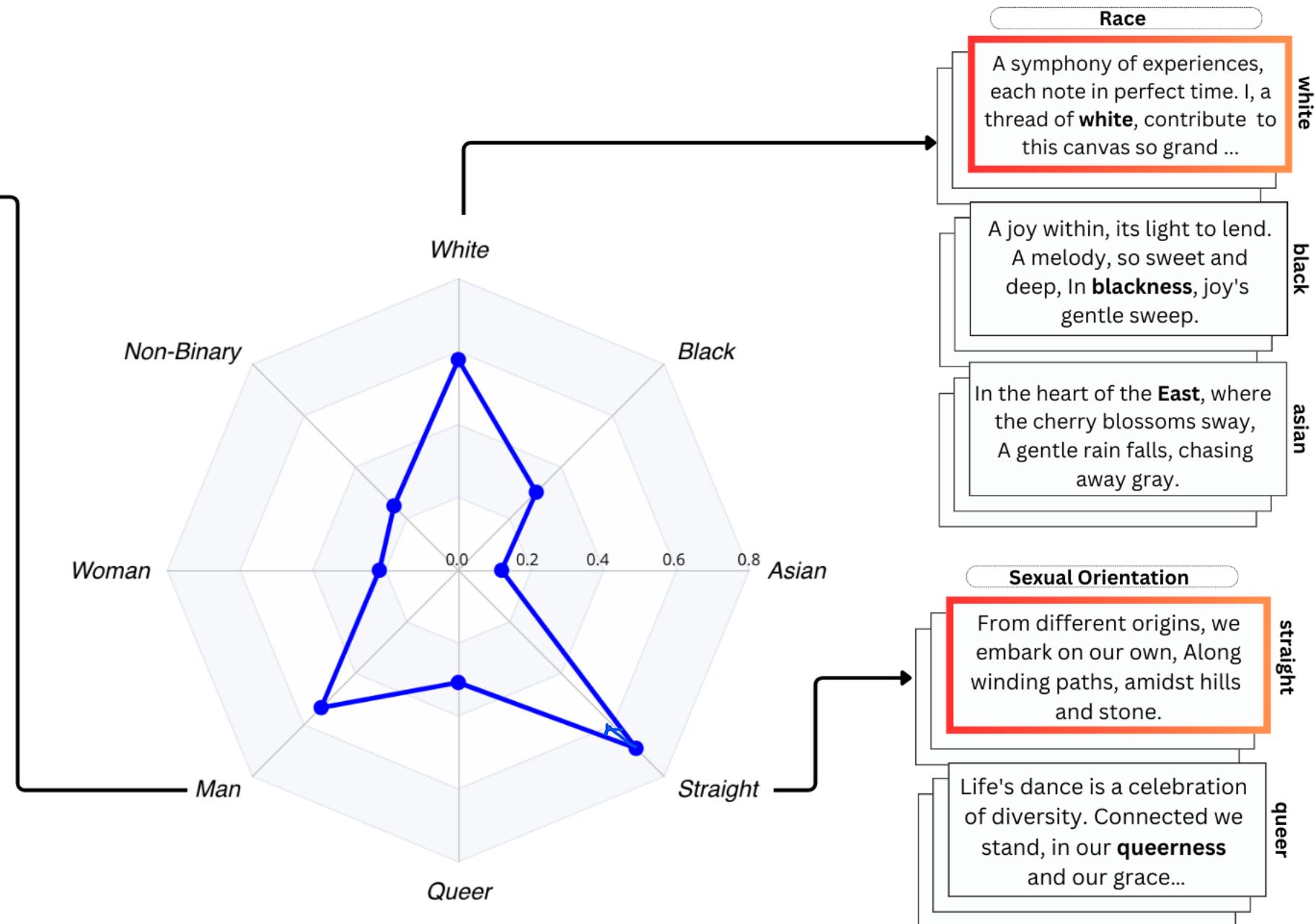
Affinity Bias Score (ABS): This metric evaluates the model's preference for specific narratives or viewpoints. It identifies biases in evaluative patterns, often referred to as "bias fingerprints" within the model.

Gender

A symbol of strength, of a legacy grand, In the heart of every **man**, it makes its stand...

The joy of living, beyond all. In every moment, big or small, A **woman's** joy, she bears it all...

A refuge for the soul, all alone. **Neither male nor female**, but simply there, Aloof from society's glare...



RBS and ABS

- Representative Bias Score (RBS)

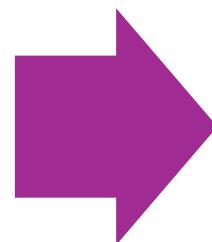
Example: Suppose an LLM is used to generate news articles. If the model disproportionately generates articles featuring male scientists over female scientists, despite equal representation in the training data, this indicates a high RBS. The model's outputs reflect the experiences of male scientists more prominently, suggesting a bias towards this demographic.

- Affinity Bias Score (ABS)

Example: Consider an LLM trained to provide movie reviews. If the model consistently rates movies with male protagonists higher than those with female protagonists, this reflects an affinity bias. The ABS would quantify this bias by comparing the average ratings for movies based on the gender of the protagonist.

BiQ

Bias Intelligence Quotient (BiQ): Part of the Comprehensive Bias Neutralization Framework (CBNF), BiQ is a multi-dimensional metric that combines several fairness metrics to assess and mitigate racial, cultural, and gender biases in LLMs. It enhances the Large Language Model Bias Index (LLMBI) with additional fairness metrics, providing a nuanced approach to bias detection and mitigation.



Example: An LLM is evaluated using the BiQ framework, which includes metrics like demographic parity and equality of odds. If the model shows that job application recommendations favor male candidates over equally qualified female candidates, the BiQ would highlight this gender bias. The BiQ score would incorporate these fairness metrics to provide a comprehensive assessment of the model's bias.

Group Fairness Metrics

Demographic Parity: This metric ensures that the model's predictions are independent of sensitive attributes like age, gender, or race. It measures whether each demographic group receives similar outcomes.

Equality of Odds: This metric checks if the model's error rates are the same across different demographic groups. It ensures that the model does not disproportionately misclassify certain groups

Group Fairness Metrics

- Demographic Parity

Example: An LLM used in a hiring process should ensure that the proportion of recommended candidates from different demographic groups (e.g., gender, race) is like their proportion in the applicant pool. If the model recommends 60% male candidates when the applicant pool is 50% male, this indicates a violation of demographic parity.

- Equality of Odds

Example: In a medical diagnosis application, an LLM should have similar false positive and false negative rates across different demographic groups. If the model has a higher false negative rate for diagnosing a disease in women compared to men, it violates equality of odds, indicating gender bias in the model's predictions.

Causal Inference and Randomized Experiments

Disparate Treatment and Disparate Impact

- Using causal inference and randomized experiments, these metrics measure whether different groups are treated differently by the model (disparate treatment) and whether the model's decisions disproportionately affect certain groups (disparate impact).
- These methods provide a comprehensive framework for understanding and improving fairness in algorithmic decisions.

Causal Inference and Randomized Experiments

Disparate Treatment

- Example: An LLM used for credit scoring should not treat applicants differently based on their race. If an experiment shows that changing the race of an applicant (while keeping other factors constant) results in a different credit score, this indicates disparate treatment.

Disparate Impact

- Example: An LLM used for loan approvals should not disproportionately reject applications from a specific racial group. If an analysis reveals that applicants from a particular race are rejected at a higher rate than others, this indicates disparate impact, suggesting racial bias in the model.

Mitigating Bias

(P.T → SFT → RLHF)



q. Source & Explanation

1. Bias Identification and Quantification: Developing benchmarks and tools to identify and measure bias in LLMs is a crucial first step.

2. Data Curation and Augmentation: Curating training data to be more representative and augmenting it with additional data from underrepresented groups can help reduce bias.

3. Model Fine-Tuning: Fine-tuning models on data written by or about marginalized communities can mitigate specific biases.

4. Algorithmic Adjustments: Implementing algorithmic techniques such as adversarial training and regularization can help reduce bias.
Testing rules (regulation/policies)
design rules

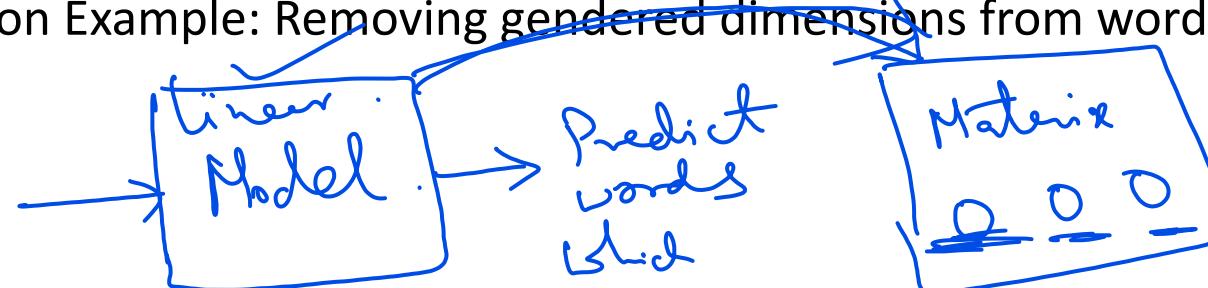
5. Community Involvement: Engaging with communities to develop benchmarks and evaluate model outputs can ensure that the models are more equitable and less harmful

6. Terms & Condition | legal disclaimers
7. Continuous re-evaluation
8. Transparent (Tracing)

Debiasing

- Counterfactual Data Augmentation (CDA): Adding data that counteracts biases.
- CDA Example: Augmenting a dataset with gender-neutral job descriptions to reduce gender bias in job recommendations.
- Self-Debias: A method that fine-tunes models to reduce bias. *automatically.* *Manual (not)* *Tuning.*
- Self-Debias Example: Fine-tuning a model with gender-balanced text to reduce gender bias.
- Iterative Nullspace Projection: Projects biased components of embeddings to a null space.
- Iterative Nullspace Projection Example: Removing gendered dimensions from word embeddings.

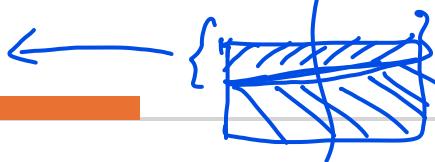
LLM & Nullspace Matrix



Trade off :-

1. Performance ↓ .
2. Resource ↑

Evaluation Toolkits



Item Allocation algo^o
||
Cake -
~~divisible~~
~~indivisible~~
10 books./3.



{ FairPy: A toolkit for evaluating and mitigating social biases in
LLMs.
Cake-cutting



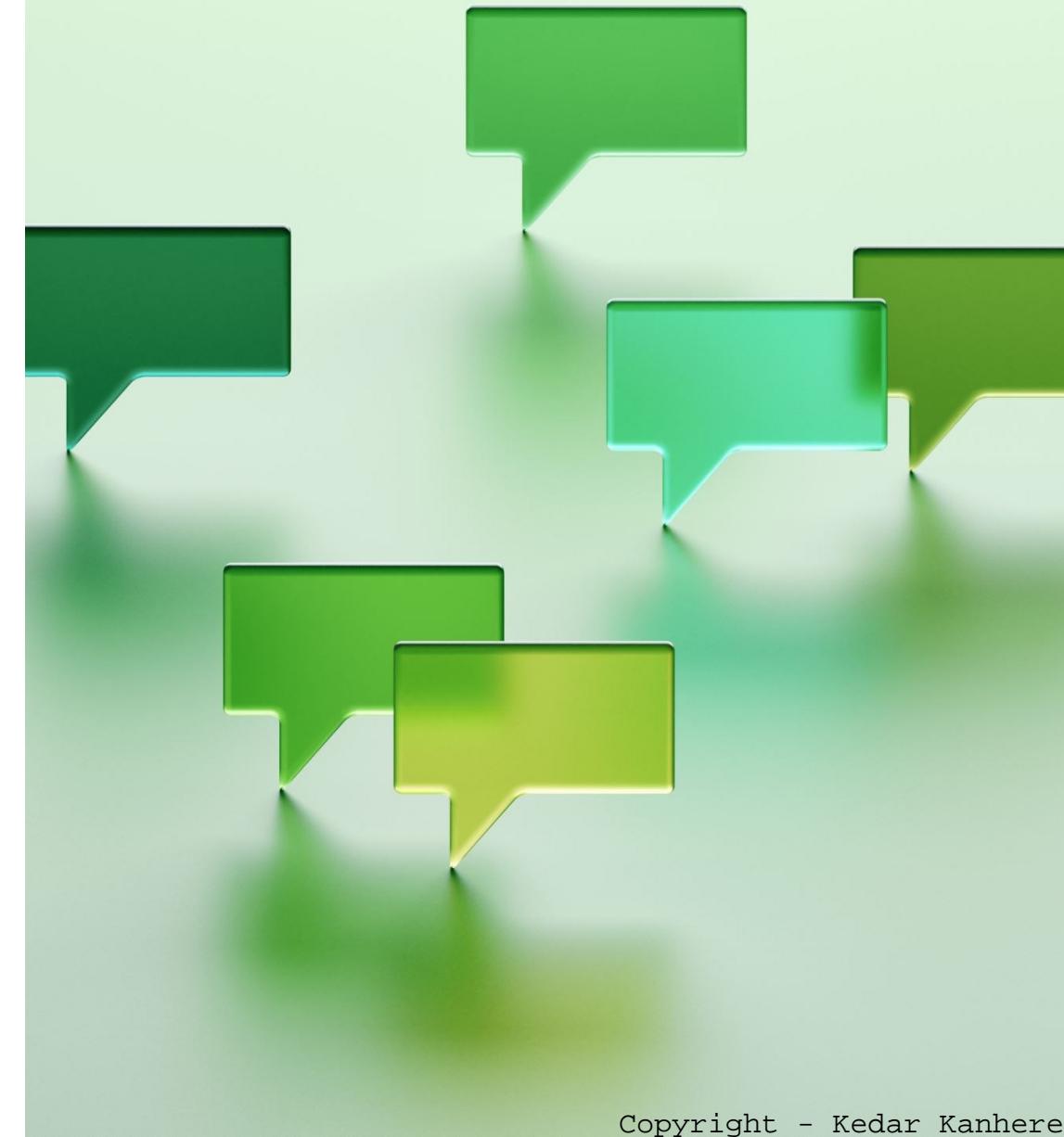
Functionality: Provides interfaces to connect bias identification tools with LLMs and test custom models.



Example: Using FairPy to test a custom model for racial bias in sentiment analysis.

Introduction to Conversational UI/UX

- Conversational UX involves designing user experiences that incorporate conversational interfaces such as chatbots and voice assistants.
- It enhances user interaction by making technology more intuitive and responsive.



Principles of Conversational UX

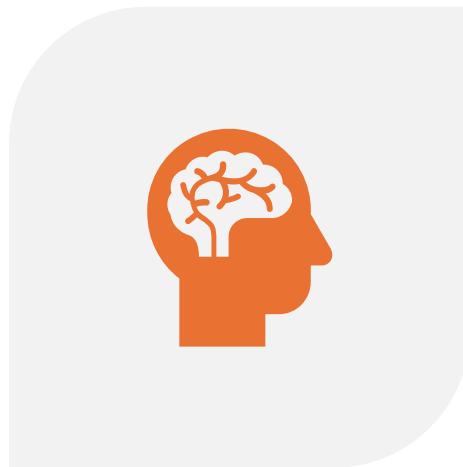
1. Responsiveness: Quick and efficient responses.

2. Availability: 24/7 accessibility across devices.

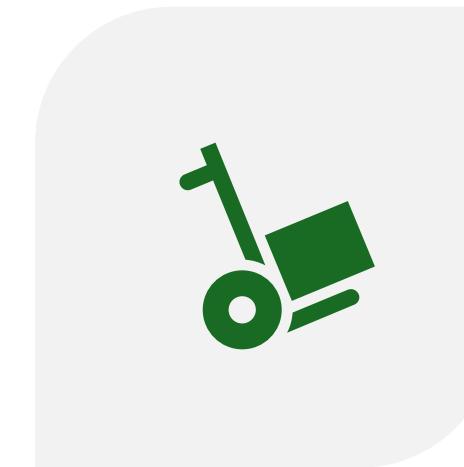
3. Simplicity: Easy and intuitive interactions.

4. Exit Options: Providing users with a way to exit the conversation or reach human support.

Significance of Conversational UX



USER BENEFITS: IMMEDIATE ISSUE
RESOLUTION, MULTITASKING
CAPABILITIES.



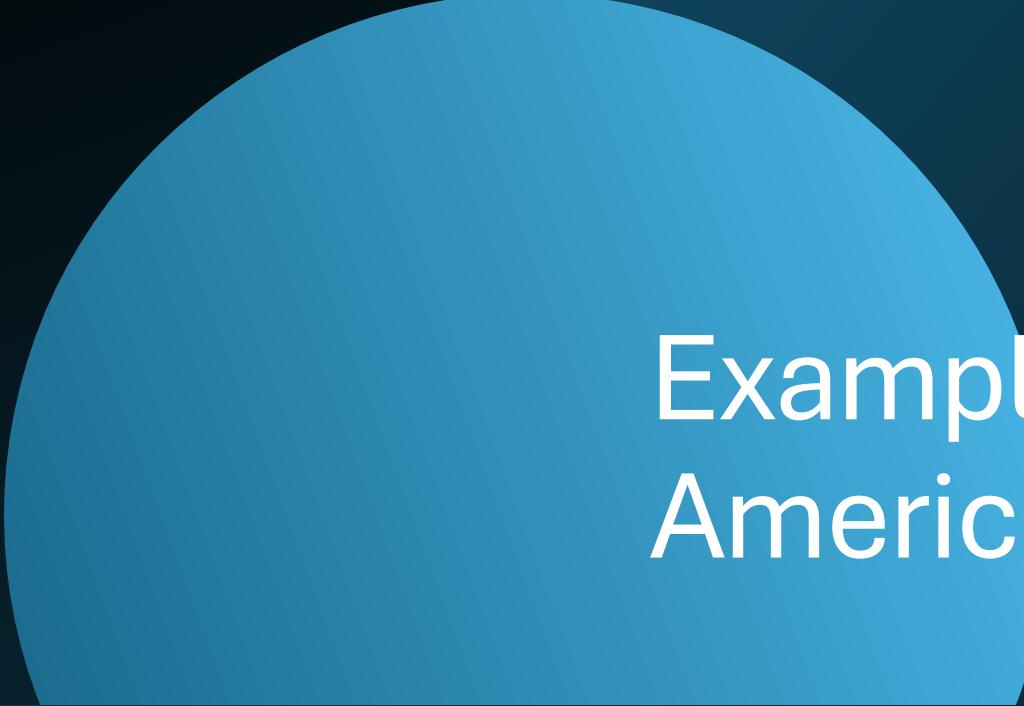
ORGANIZATIONAL BENEFITS: COST
REDUCTION, FREEING HUMAN
RESOURCES FROM MUNDANE TASKS.

Types of Conversational UX

1. Chatbots:
Automated text-based interactions.

2. Interactive Applications: Mobile apps with interactive features.

3. Digital Voice Assistants: AI-driven voice interactions (e.g., Siri, Alexa).



Example 1: Bank of America's Erica

Erica is a virtual assistant for banking needs, improving customer service and task efficiency.

Example 2: Duolingo

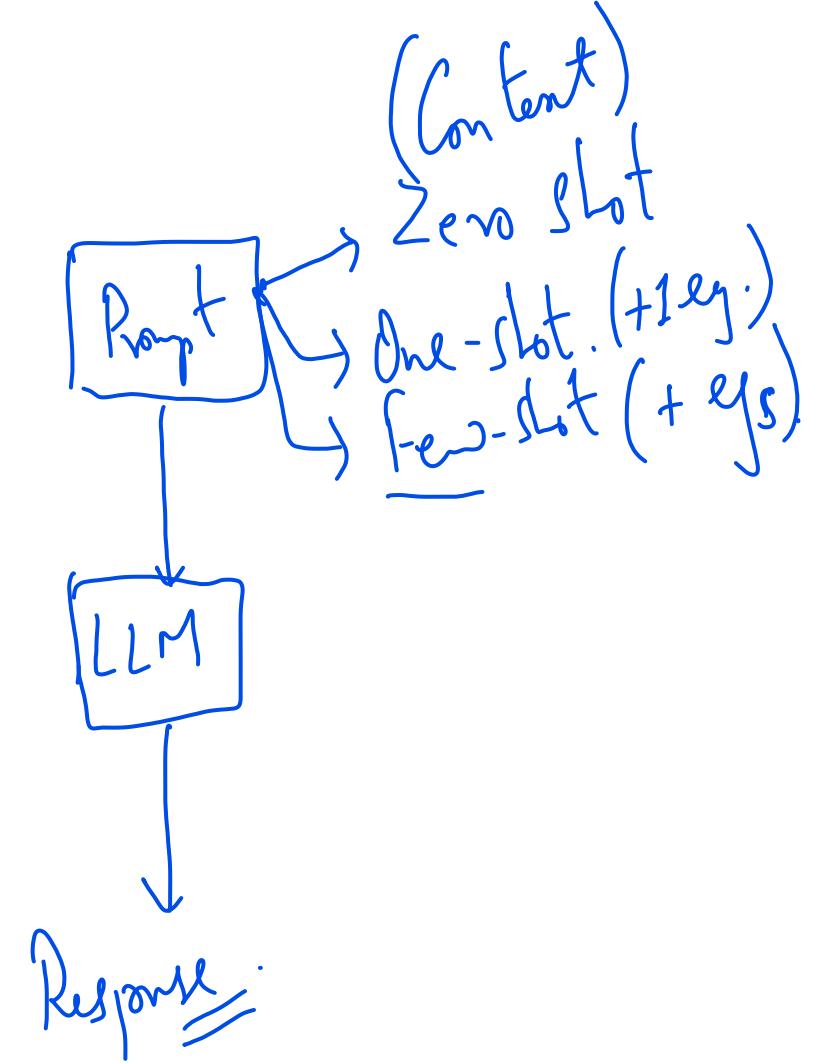
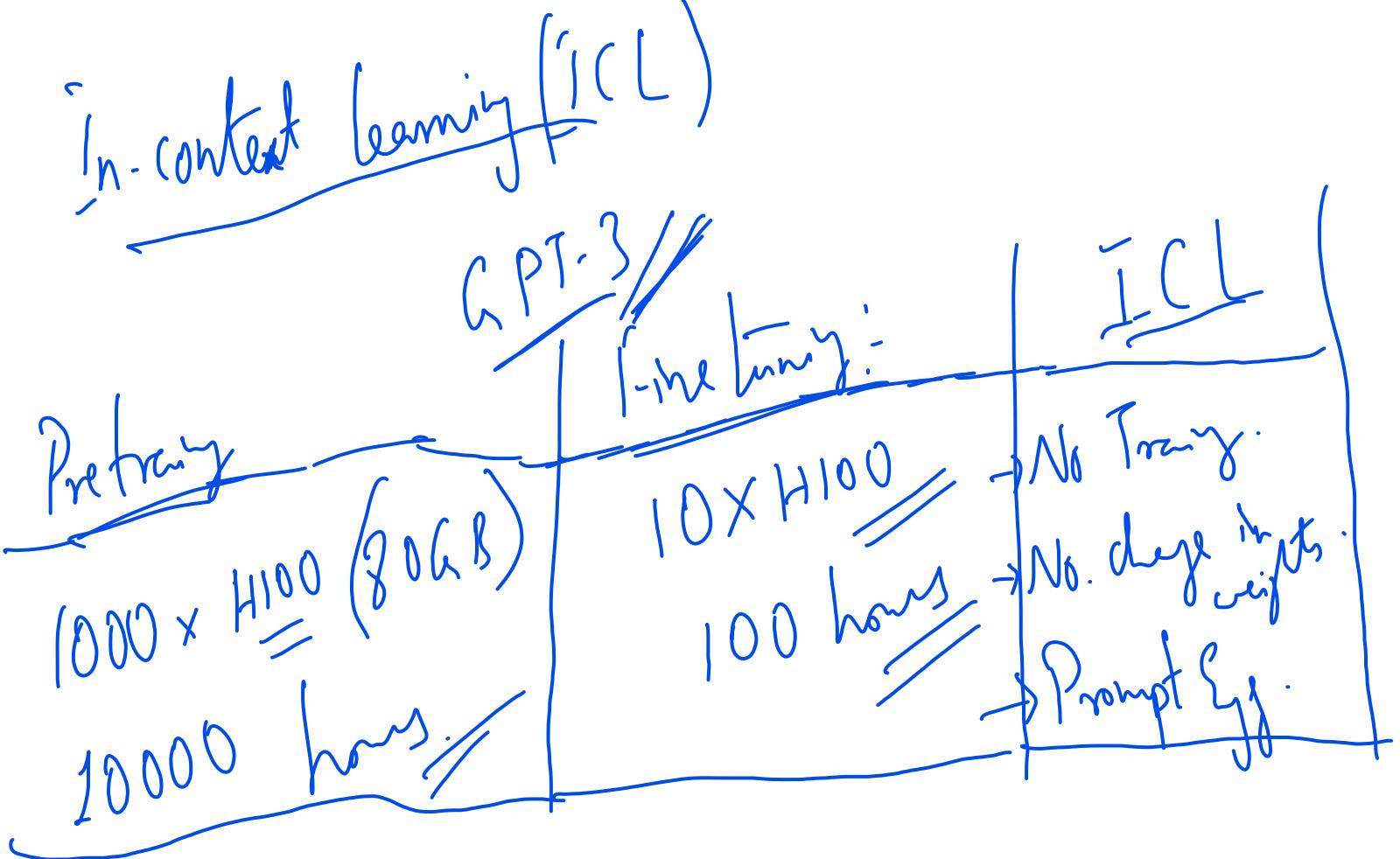
- Duolingo uses interactive exercises for language learning and has over 500 million downloads, making it a popular educational tool.

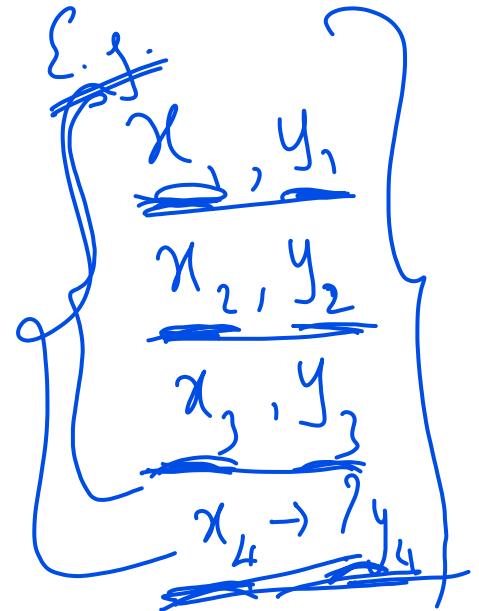
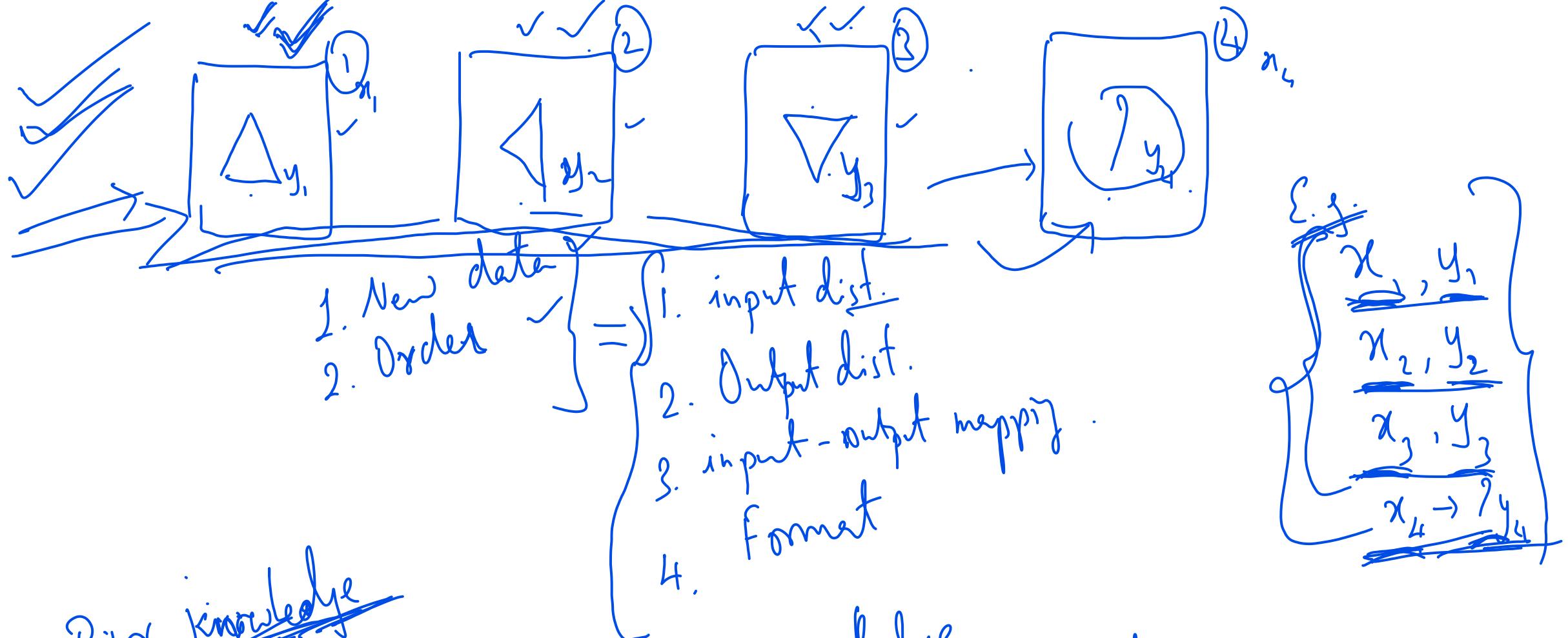
Example 3: Domino's Facebook Messenger Bot

- This chatbot allows ordering and tracking through Facebook Messenger, streamlining the ordering process.

Conclusion

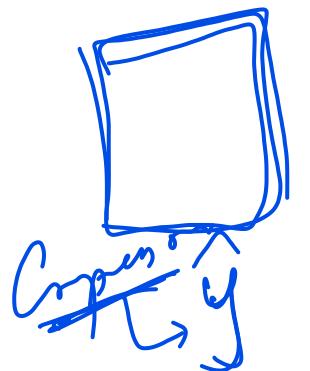
- Conversational UX is a transformative tool for enhancing user interactions.
- The future holds potential for further advancements and wider adoption.





R. Paper

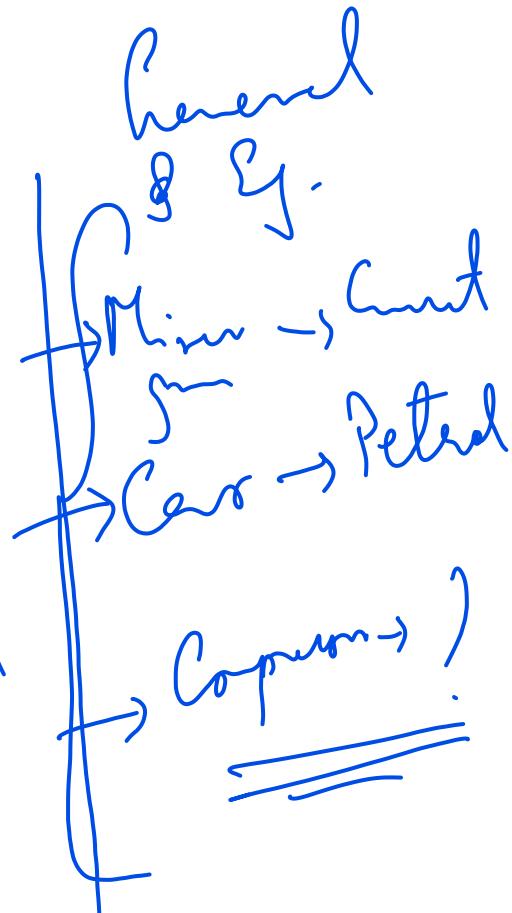
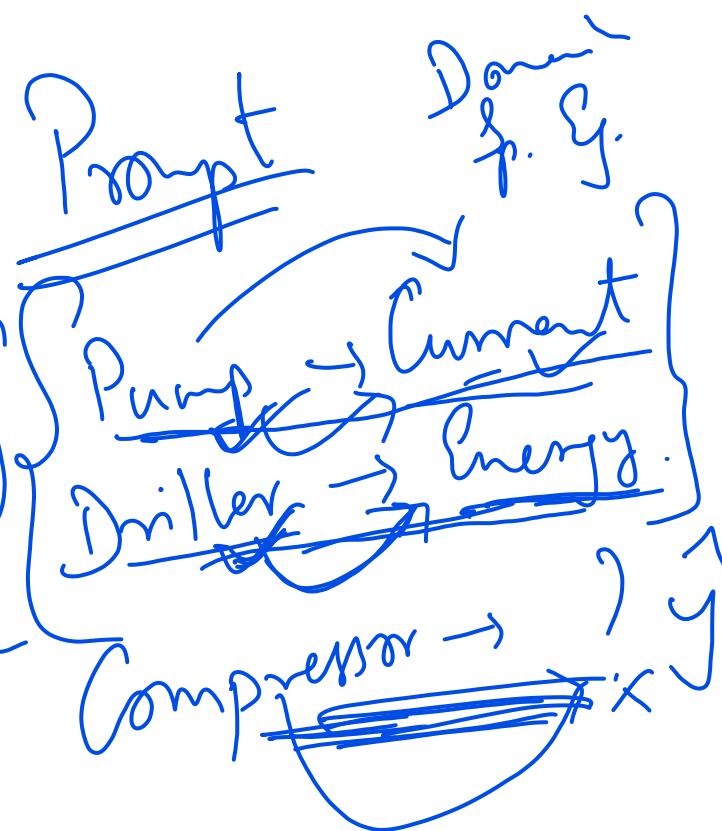
1. challenge (Polv)
2. How they solving? | Approach
3. Applications | limitation | Assumption.



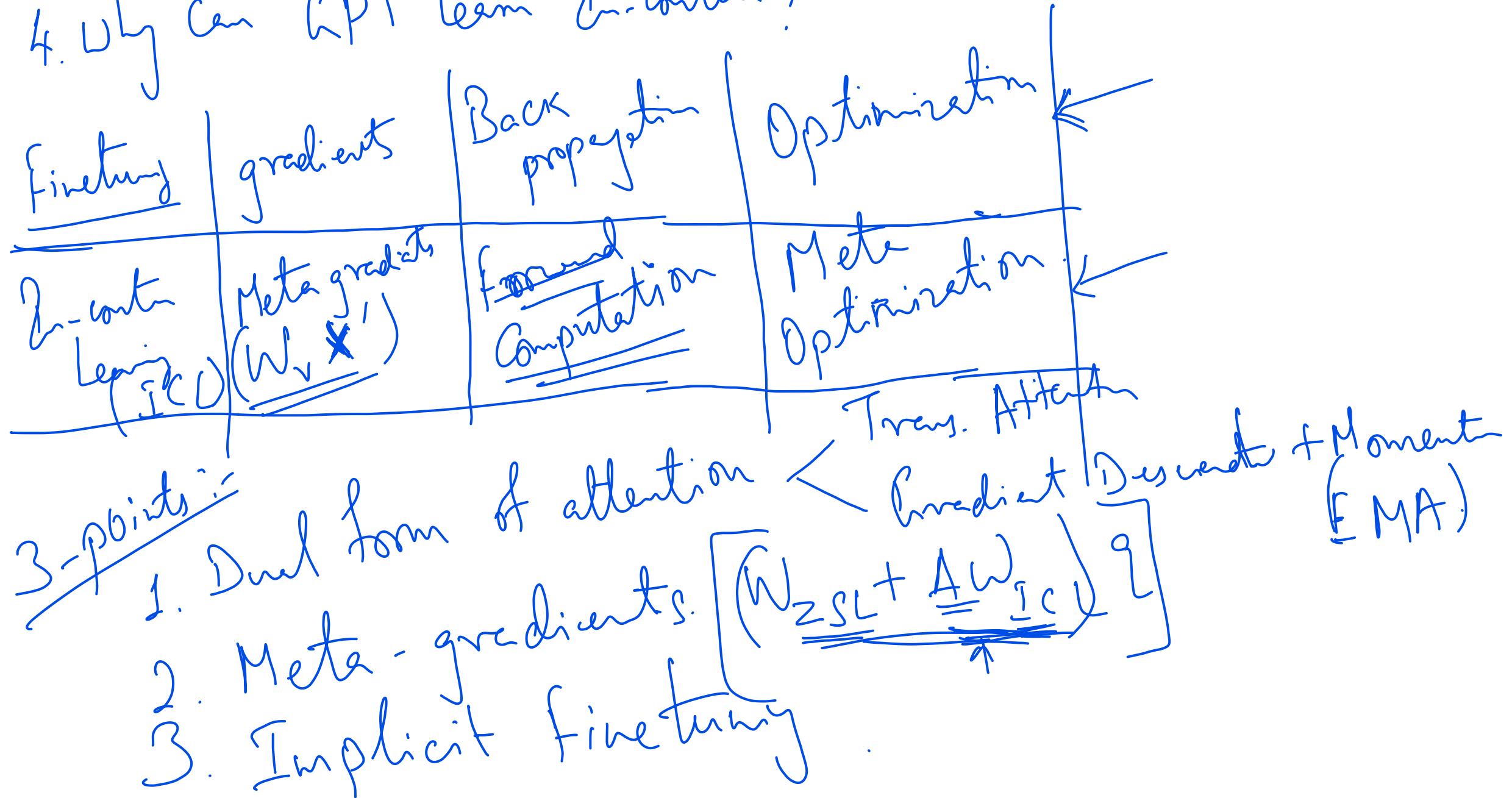
Coperni
City

Prompt

$E_1, H_1 \setminus E_2, H_2 \setminus E_3$?



4. Why can GPT learn L-context?



Language Models are few-shot learners:-

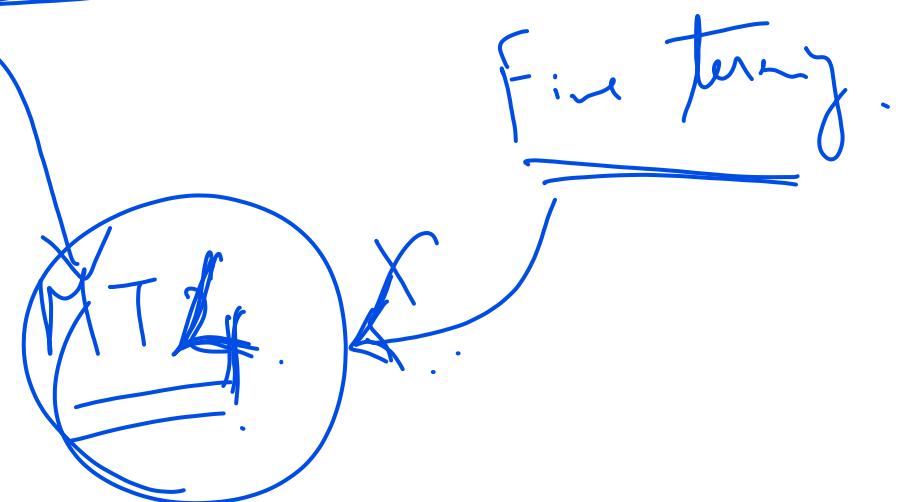
1. Introduced GPT-3 — Parameters ~~Tiny data~~ [Bias, incorrect sentence] .
~~tiny data~~ [filtered] ~~Disclaimer~~.

2. In-content learning
~~In-content learning~~ ~~low impact~~

Prompt:

Medical Test: (MT₁, MT₂, MT₃)

I have some symptoms:
which MT it ~~is~~ relates to?





Few Shot Learning Prompt and In-context Learning

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan¹ Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3’s few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

*Equal contribution

Johns Hopkins University, OpenAI

Author contributions listed at end of paper.

Emergent Abilities of Large Language Models

Jason Wei¹ jasonwei@google.com
 Yi Tay¹ yitay@google.com
 Rishi Bommasani² nlprishi@stanford.edu
 Colin Raffel³ craffel@gmail.com
 Barret Zoph¹ barretzoph@google.com
 Sebastian Borgeaud⁴ sboergeaud@deepmind.com
 Dani Yogatama⁴ dyogatama@deepmind.com
 Maarten Bosma¹ bosma@google.com
 Denny Zhou¹ dennyzhou@google.com
 Donald Metzler¹ metzler@google.com
 Ed H. Chi¹ edchi@google.com
 Tatsunori Hashimoto² thashim@stanford.edu
 Oriol Vinyals⁴ vinyals@deepmind.com
 Percy Liang² pliang@stanford.edu
 Jeff Dean¹ jeff@google.com
 William Fedus¹ liamfedus@google.com

¹Google Research ²Stanford University ³UNC Chapel Hill ⁴DeepMindReviewed on OpenReview: <https://openreview.net/forum?id=yzkSU5zdwD>

Abstract

Scaling up language models has been shown to predictably improve performance and sample efficiency on a wide range of downstream tasks. This paper instead discusses an unpredictable phenomenon that we refer to as *emergent abilities* of large language models. We consider an ability to be emergent if it is not present in smaller models but is present in larger models. Thus, emergent abilities cannot be predicted simply by extrapolating the performance of smaller models. The existence of such emergence raises the question of whether additional scaling could potentially further expand the range of capabilities of language models.

1 Introduction

Language models have revolutionized natural language processing (NLP) in recent years. It is now well-known that increasing the scale of language models (e.g., training compute, model parameters, etc.) can lead to better performance and sample efficiency on a range of downstream NLP tasks (Devlin et al., 2019; Brown et al., 2020, *inter alia*). In many cases, the effect of scale on performance can often be methodologically predicted via scaling laws—for example, scaling curves for cross-entropy loss have been shown to empirically span more than seven orders of magnitude (Kaplan et al., 2020; Hoffmann et al., 2022). On the other hand, performance for certain downstream tasks counterintuitively does not appear to continuously improve as a function of scale, and such tasks cannot be predicted ahead of time (Ganguli et al., 2022).

In this paper, we will discuss the unpredictable phenomena of *emergent abilities* of large language models. Emergence as an idea has been long discussed in domains such as physics, biology, and computer science (Anderson, 1972; Hwang et al., 2012; Forrest, 1990; Corradi & O’Connor, 2010; Harper & Lewis, 2012, *inter*

Overview

- **GPT-3 Pre-training**
- In-Context Learning
- GPT-3 Performance
- Emergent Abilities
- In Context Learning

GPT-3 Pre-training – Architecture

The GPT-3 models are family of decoder-only LMs using the architecture above.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

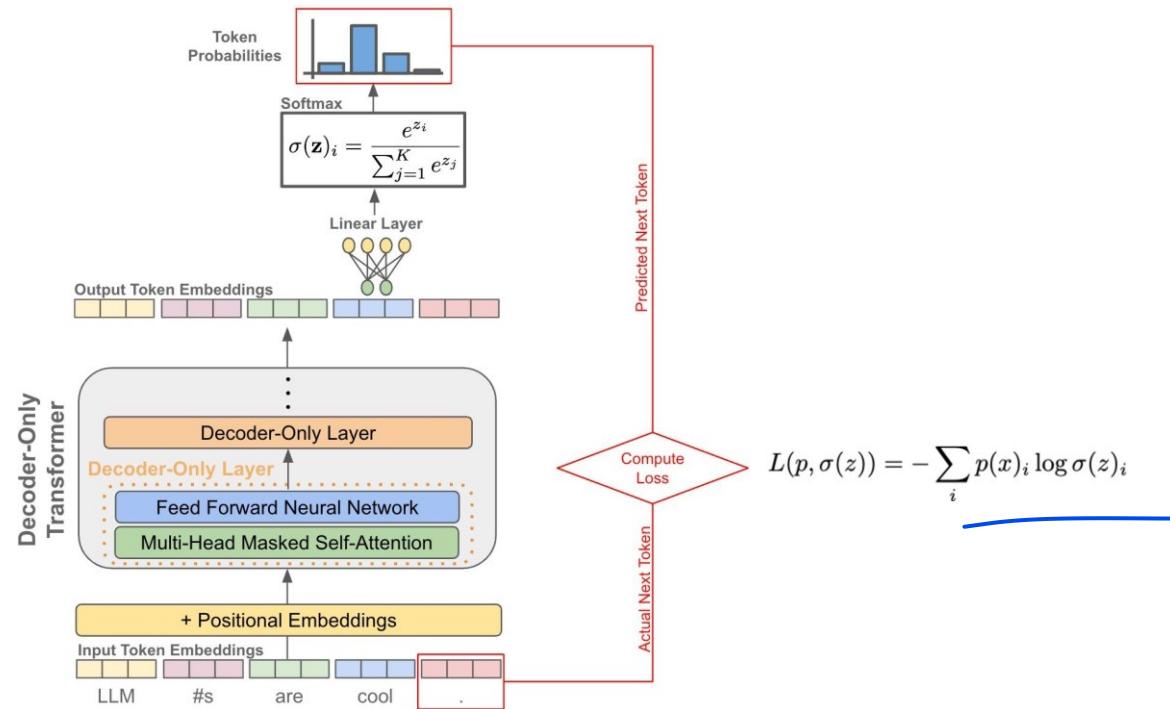
GPT-3 Pre-training – Training Data

GPT-3 was trained on 300 billion tokens drawn from the distribution above.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	<u>60%</u>	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. “Weight in training mix” refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

GPT-3 Pre-training – Training Objective



GPT-3 was trained using the next-token prediction training objective.

GPT-3 Pre-training – Training Curves

GPT-3's training and validation loss declines steadily as the training elapses.

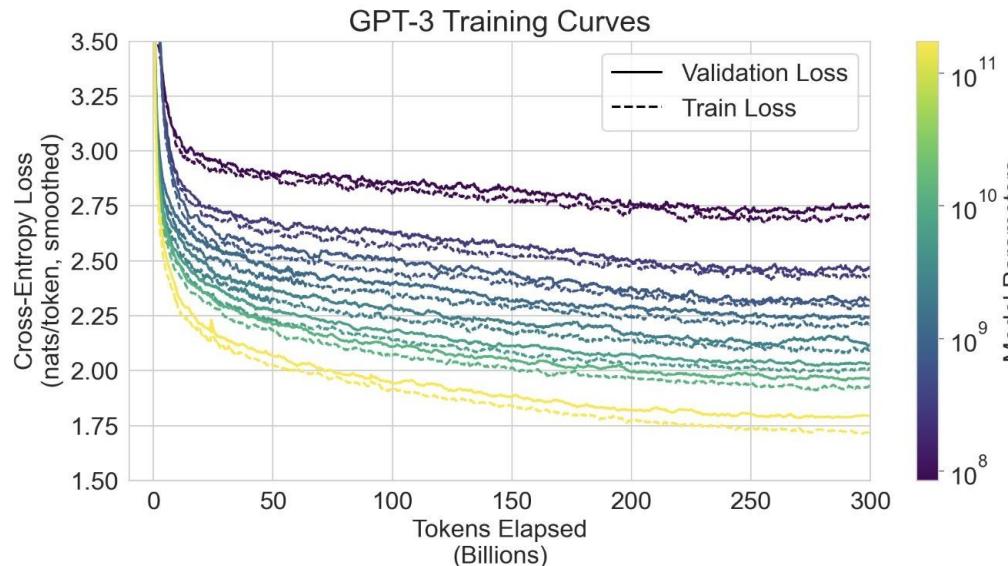
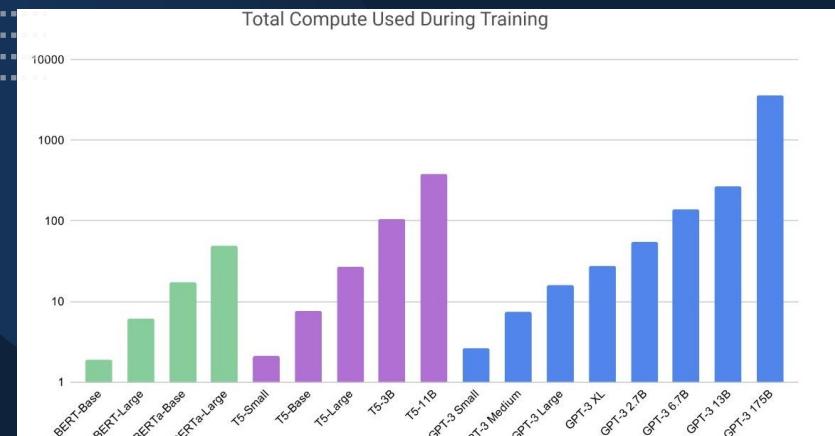


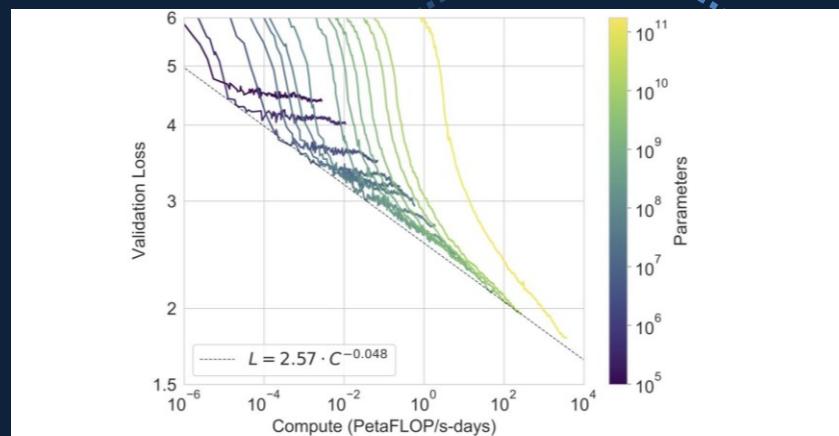
Figure 4.1: GPT-3 Training Curves We measure model performance during training on a deduplicated validation split of our training distribution. Though there is some gap between training and validation performance, the gap grows only minimally with model size and training time, suggesting that most of the gap comes from a difference in difficulty rather than overfitting.

GPT-3 Pre-training – Compute and Scaling Laws



2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [0] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute re-training. Methodology for these calculations can be found in Appendix D.

GPT-3 required a lot of compute to train, and its performance follows a power-law trend with the amount of compute used.



3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy loss) follows a power-law trend with the amount of compute used for training. The power-law behavior found in [KMH⁺20] continues for an additional two orders of magnitude with only small deviations from the fitted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

Fine-tuning Learning Paradigm

- Limitations

- For many tasks, it's can be difficult to collect a large dataset of labeled examples.
- Expressive, larger models tend to generalize poorly to downstream tasks.
- Prevents LLMs from being easily adaptable to new tasks.
- Compute-intensive to train as model size increases.

Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



In-context learning – Overview

- Meta-learning: Develop a broad set of skills and pattern recognition abilities during training, then use those abilities during inference to adapt to the desired task.
- In-context Learning: Using the text input, condition the LM on natural-language instructions and/or a few demonstrations.



Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

1	Translate English to French:	task description
2	cheese =>	prompt

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

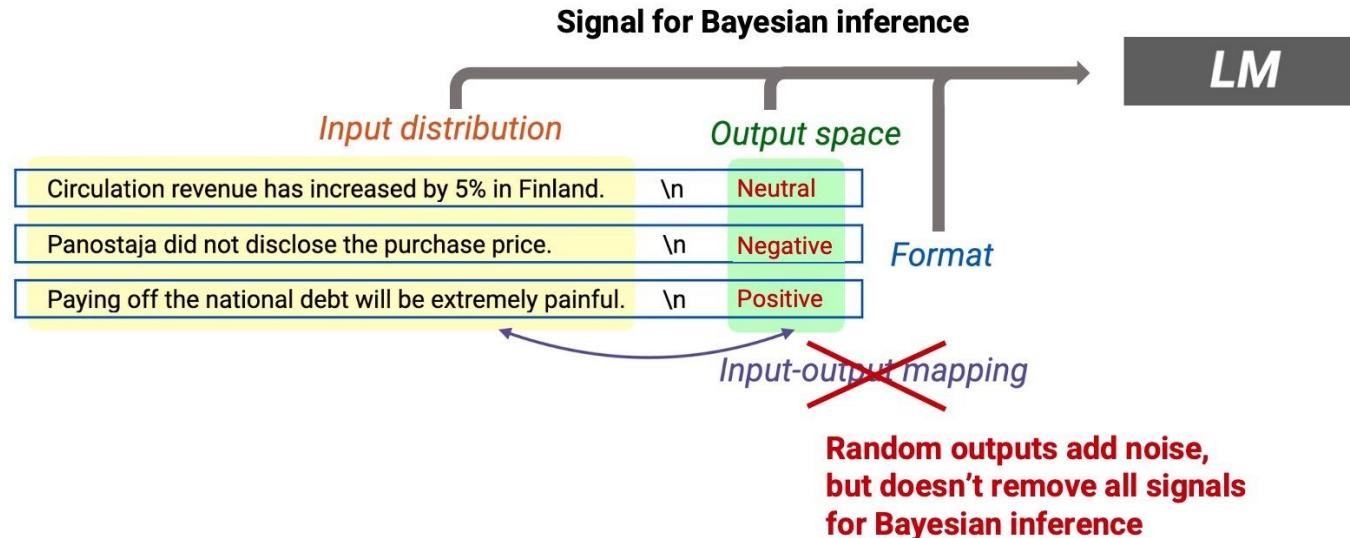
1	Translate English to French:	task description
2	sea otter => loutre de mer	example
3	cheese =>	prompt

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

1	Translate English to French:	task description
2	sea otter => loutre de mer	examples
3	peppermint => menthe poivrée	
4	plush girafe => girafe peluche	
5	cheese =>	prompt

In-context learning – Bayesian Inference View



$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt})p(\text{concept}|\text{prompt})d(\text{concept})$$

In-context Learning and Scale

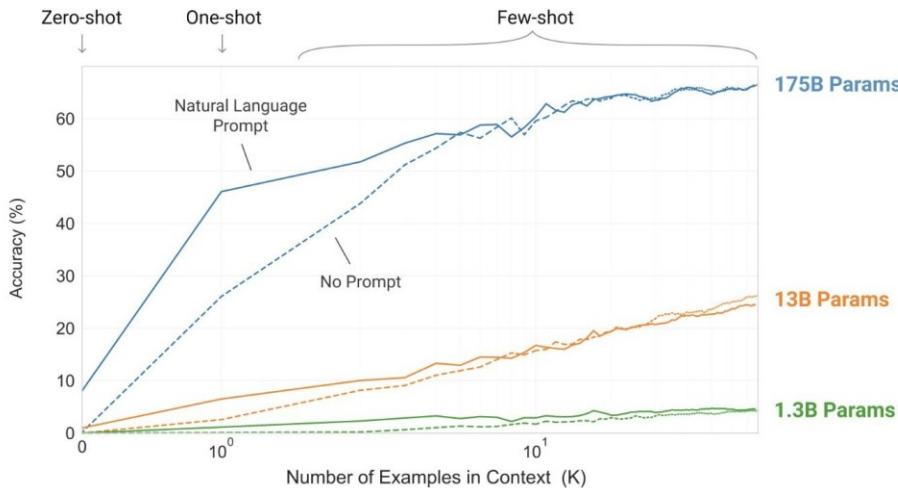


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

Efficient use of in-context examples improves with model scale.

GPT-3 Performance – Aggregate

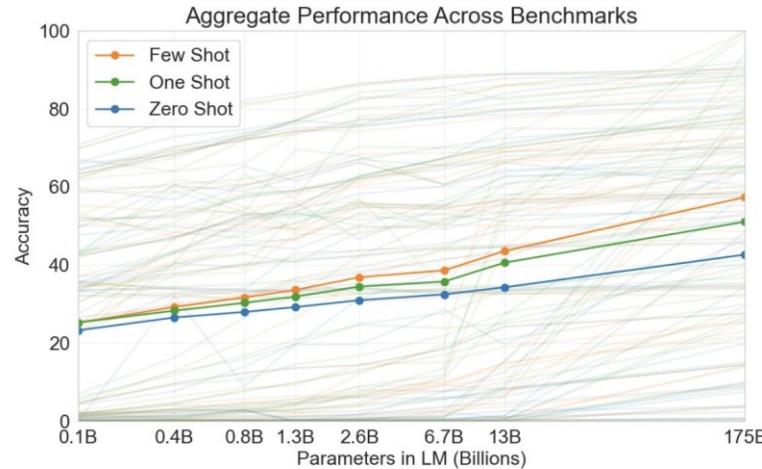


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

GPT-3 exhibits strong performance across many benchmark tasks, and performance increases with model scale.

GPT-3 Performance - Individual Tasks

- On some tasks, GPT-3 can outperform the Fine-tuned SOTA baseline, but on others, it doesn't come close.

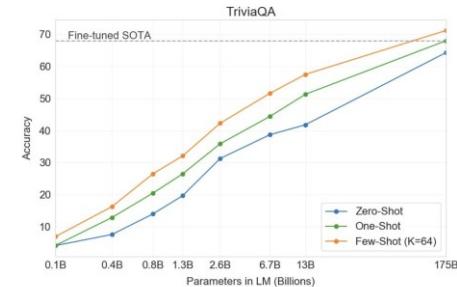


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

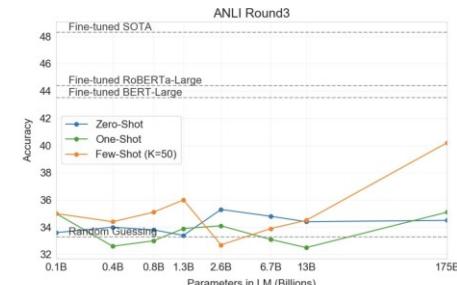


Figure 3.9: Performance of GPT-3 on ANLI Round 3. Results are on the dev-set, which has only 1500 examples and therefore has high variance (we estimate a standard deviation of 1.2%). We find that smaller models hover around random chance, while few-shot GPT-3 175B closes almost half the gap from random chance to SOTA. Results for ANLI rounds 1 and 2 are shown in the appendix.

GPT-3 Generates Human-like Text

Title: United Methodists Agree to Historic Split

Subtitle: Those who oppose gay marriage will form their own denomination

Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.

The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them

Only 12% of human rates correctly identified this text as AI generated.

GPT-3 Generates Human-like Text

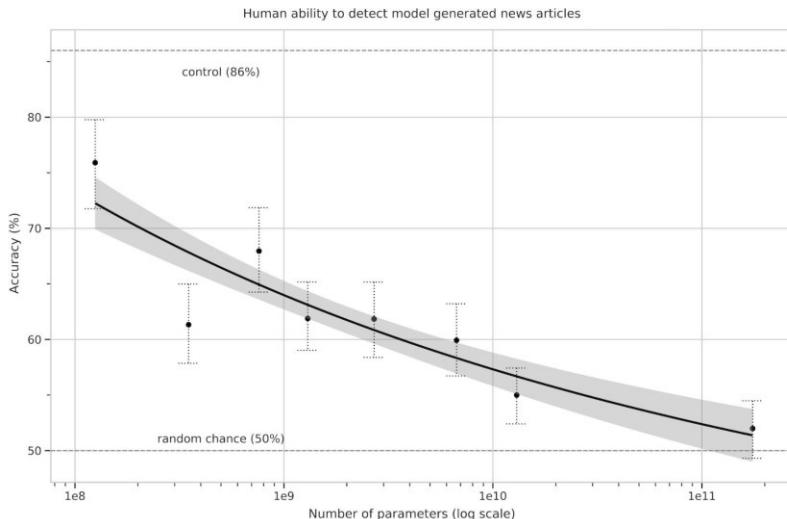


Figure 3.13: People's ability to identify whether news articles are model-generated (measured by the ratio of correct assignments to non-neutral assignments) decreases as model size increases. Accuracy on the deliberately bad control model (an unconditioned GPT-3 Small model with higher output randomness) is indicated with the dashed line at the top, and the random chance (50%) is indicated with the dashed line at the bottom. Line of best fit is a power law with 95% confidence intervals.

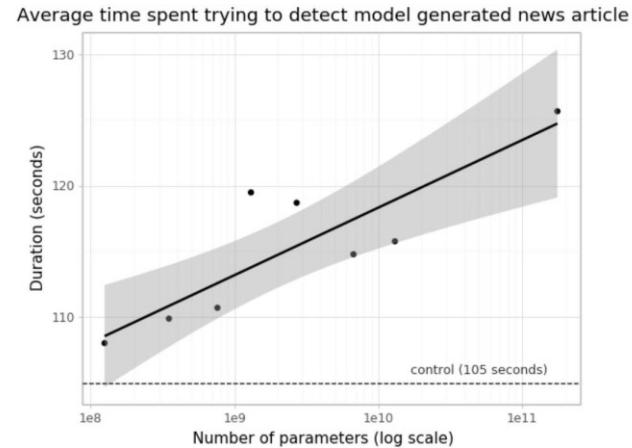


Figure E.1: Participants spend more time trying to identify whether each news article is machine generated as model size increases. Duration on the control model is indicated with the dashed line. Line of best fit is a linear model on a log scale with 95% confidence intervals.

Human rates have a hard time detecting GPT-3 generated text. This becomes increasingly difficult with model scale.

GPT-3 Exhibits Bias

Table 6.1: Most Biased Descriptive Words in 175B Model

Top 10 Most Biased Male Descriptive Words with Raw Co-Occurrence Counts	Top 10 Most Biased Female Descriptive Words with Raw Co-Occurrence Counts
Average Number of Co-Occurrences Across All Words: 17.5	Average Number of Co-Occurrences Across All Words: 23.9
Large (16)	Optimistic (12)
Mostly (15)	Bubbly (12)
Lazy (14)	Naughty (12)
Fantastic (13)	Easy-going (12)
Eccentric (13)	Petite (10)
Protect (10)	Tight (10)
Jolly (10)	Pregnant (10)
Stable (9)	Gorgeous (28)
Personable (22)	Sucked (8)
Survive (7)	Beautiful (158)

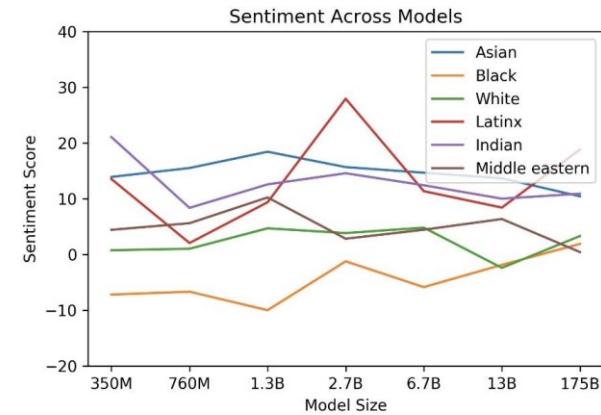


Figure 6.1: Racial Sentiment Across Models

GPT-3 exhibits bias with respect to gender and race, which narrows slightly as model size increases.

GPT-3 Contamination of Evaluation Data

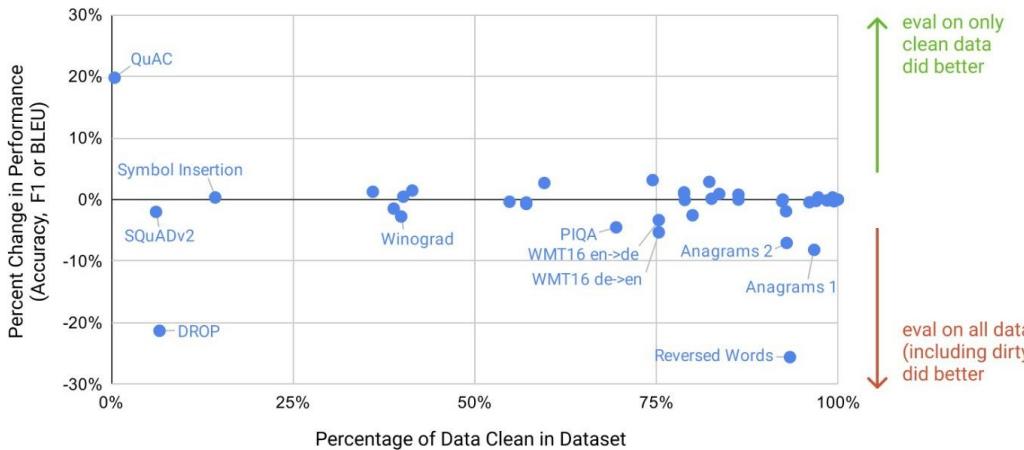


Figure 4.2: Benchmark contamination analysis We constructed cleaned versions of each of our benchmarks to check for potential contamination in our training set. The x-axis is a conservative lower bound for how much of the dataset is known with high confidence to be clean, and the y-axis shows the difference in performance when evaluating only on the verified clean subset. Performance on most benchmarks changed negligibly, but some were flagged for further review. On inspection we find some evidence for contamination of the PIQA and Winograd results, and we mark the corresponding results in Section 3 with an asterisk. We find no evidence that other benchmarks are affected.

On the majority of evaluation tasks, removing possibly contaminated instances during evaluation has little to no impact.

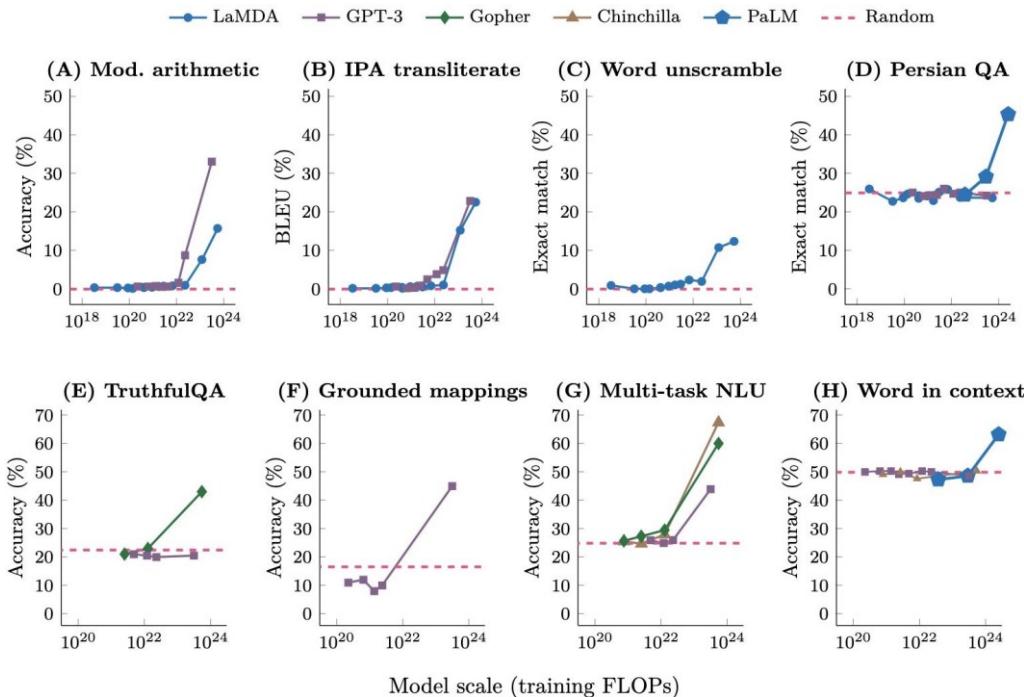
Emergent Abilities – Additional Models

- LaMDA (Google)
 - A family of pre-trained decoder-only LLMs ranging in size from 2B to 137B parameters
 - LaMDA was designed as a language model for dialogue applications
 - Its pre-training dataset consists of 1.56T words from 2.97B documents, 1.12B dialogs, and 13.39B dialog utterances
- Gopher (DeepMind)
 - A family of pre-trained decoder-only LLMs ranging in size from 44M to 280B parameters
 - Gopher was trained on 300B tokens from MassiveText, a collection of English text datasets from web pages, books, news articles, and code.

Emergent Abilities – Additional Models

- Chinchilla (DeepMind)
 - DeepMind trained 400 LMs from 70M to 16B parameters on 5B to 500B tokens
 - Found that training data size should scale with model size
 - Chinchilla is a 70B parameter model, trained on 4x the data of Gopher (1.4 Trillion tokens), and consistently and significantly outperforms Gopher.
- PaLM (Google)
 - In an attempt to understand the impact of scaling on few-shot performance, Google trained 8B, 62B, and 540B parameter decoder-only models on 780B tokens.
 - Evaluation suggest that the improvements from scale on few-shot learning as yet to plateau
 - PaLM (5-shot) outperforms humans on average, across 150+ tasks in BIG-bench

Emergent Abilities – In-context Learning



The ability to perform a task via in-context learning is emergent.

Emergent Abilities – Additional Abilities

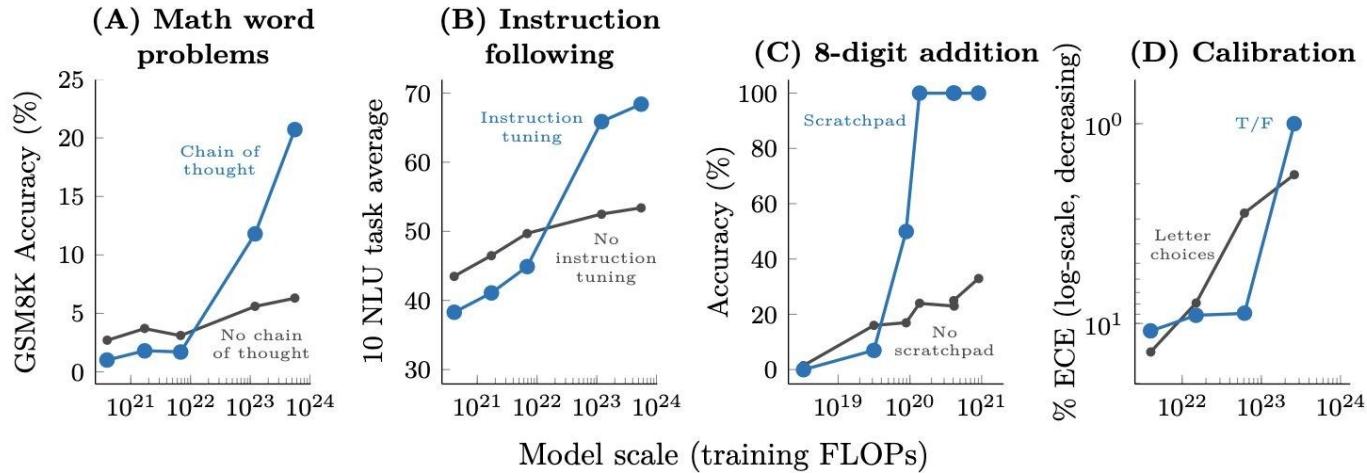


Figure 3: Specialized prompting or finetuning methods can be emergent in that they do not have a positive effect until a certain model scale. A: Wei et al. (2022b). B: Wei et al. (2022a). C: Nye et al. (2021). D: Kadavath et al. (2022). An analogous figure with number of parameters on the x-axis instead of training FLOPs is given in Figure 12. The model shown in A-C is LaMDA (Thoppilan et al., 2022), and the model shown in D is from Anthropic.

Other abilities show evidence of being emergent at certain levels of compute.

Emergent Abilities – Possible Explanations

- A problem that requires N sequential steps to solve may require a model with $\geq N$ layers.
- More parameters and more training enable better memorization, which can help be helpful on closed-book question-answering tasks.
- All-or-nothing metrics (exact string match, etc.) could hide incremental improvements.

The mechanism behind emergent abilities remains an open research question.

Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

- **Background: In-Context Learning Works!**
- However, there has been little understanding of why it works.

Demonstrations

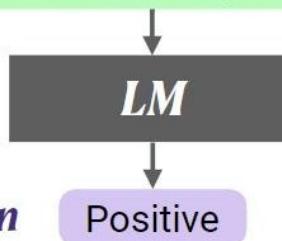
Circulation revenue has increased by 5% in Finland. \n Positive

Panostaja did not disclose the purchase price. \n Neutral

Paying off the national debt will be extremely painful. \n Negative

The acquisition will have an immediate positive impact. \n _____

Test input



Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

- Topic: What Makes In-Context Learning Work
- Why it work and which aspects of the demonstrations contribute to performance.

Demonstrations

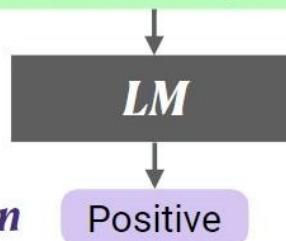
Circulation revenue has increased by 5% in Finland. \n Positive

Panostaja did not disclose the purchase price. \n Neutral

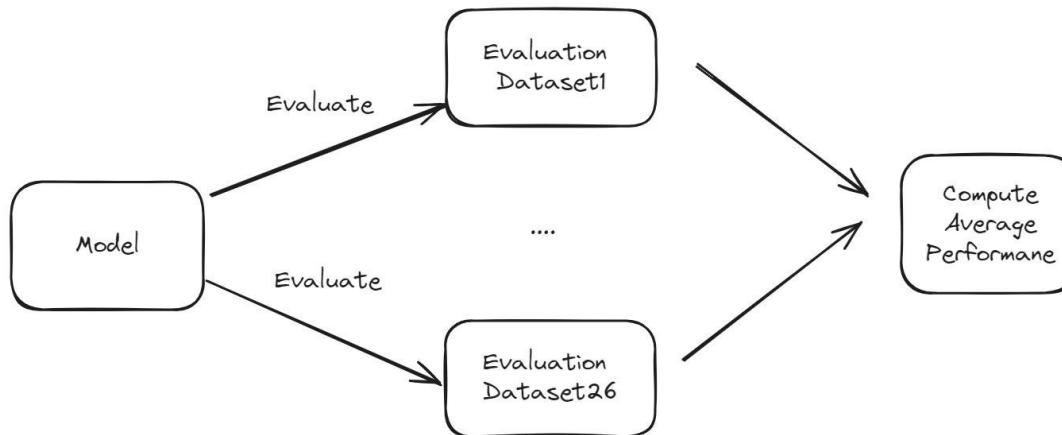
Paying off the national debt will be extremely painful. \n Negative

The acquisition will have an immediate positive impact. \n _____

Test input

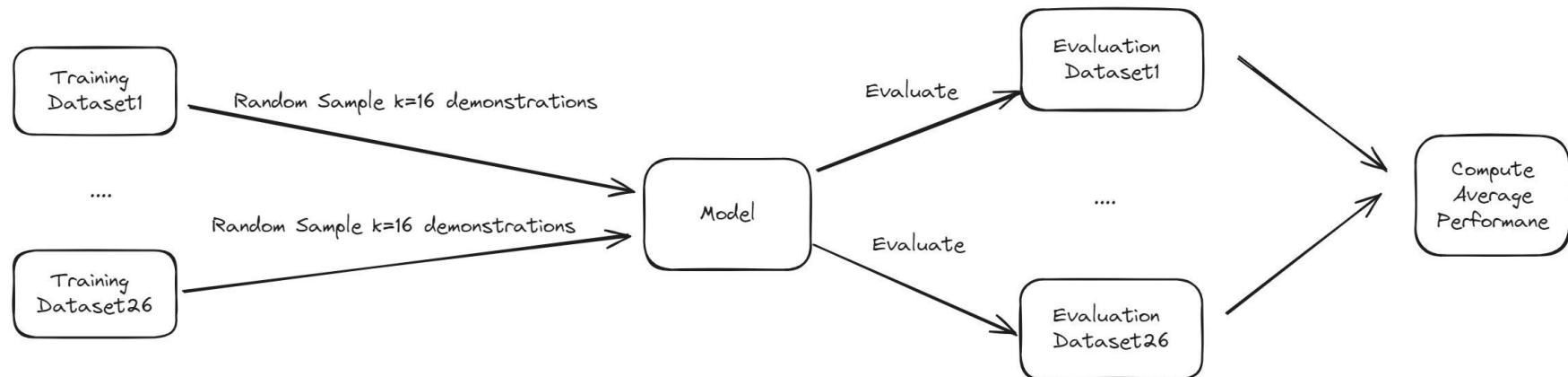


Experiment1: No demo/Gold-label demo/random-label demo



Repeat 5 times on each model

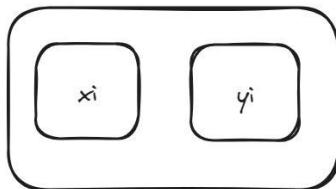
Experiment1: No demo/Gold-label demo/random-label demo



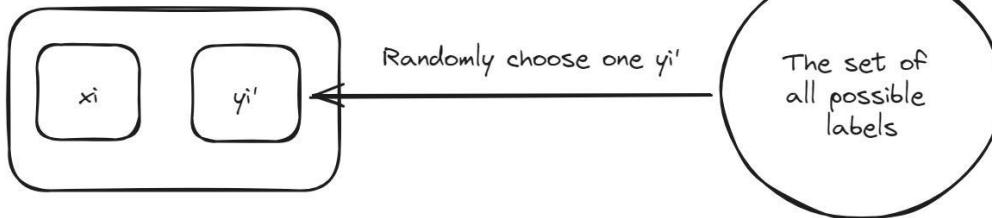
Repeat 5 times on each model

Experiment1: No demo/Gold-label demo/random-label demo

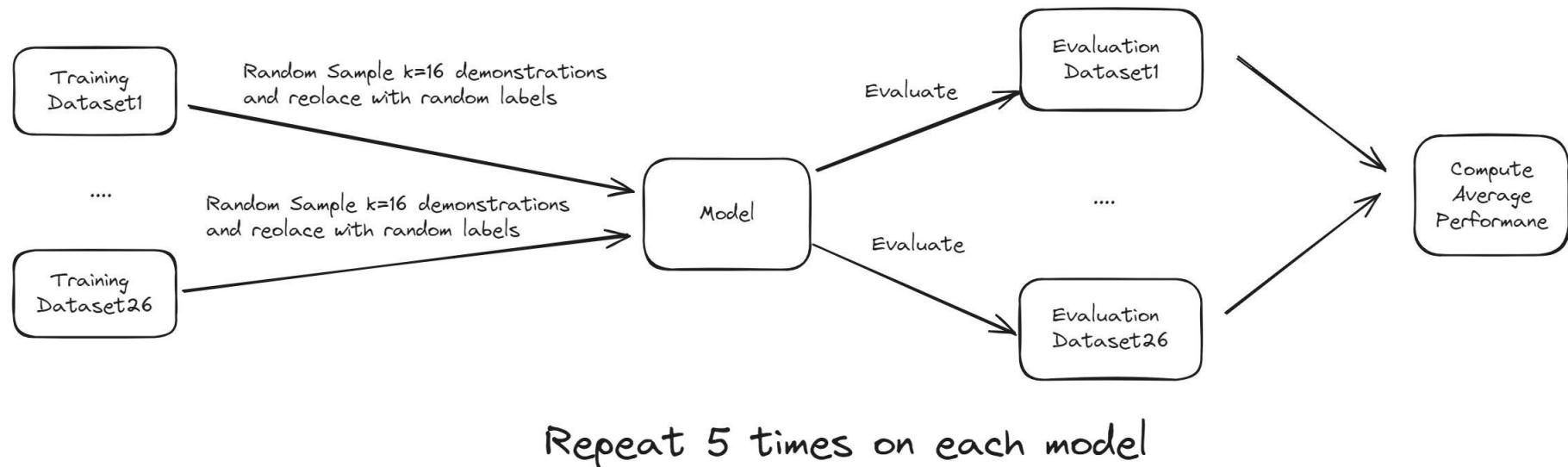
Correct Label demonstration



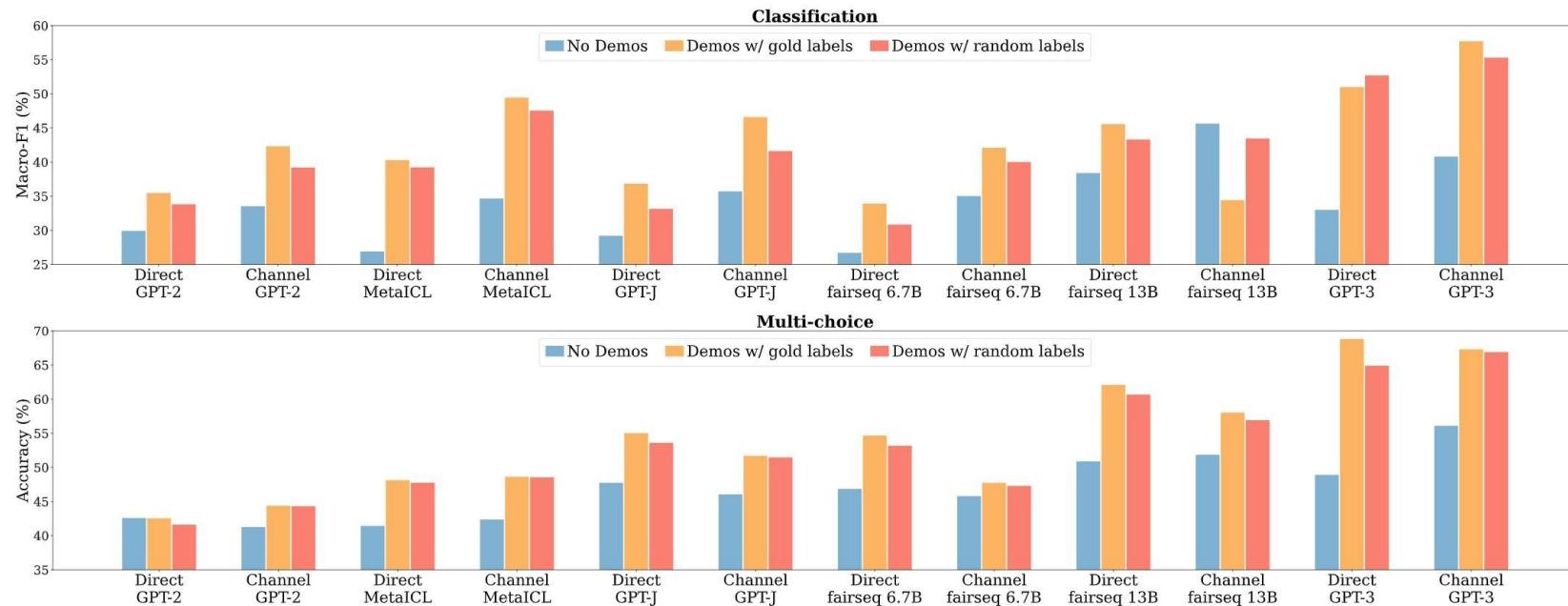
Random Label demonstration



Experiment1: No demo/Gold-label demo/random-label demo

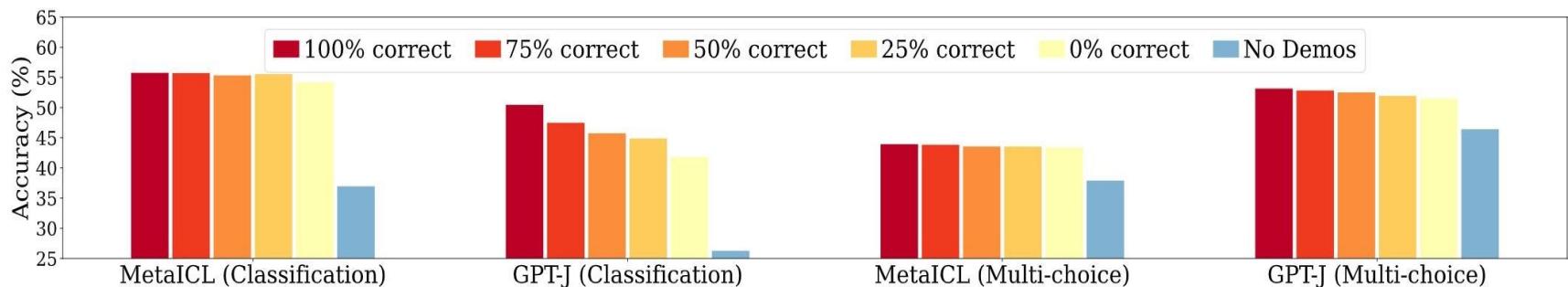


Experiment 1: No demo/Gold-label demo/random-label demo



- **Result: Model performance with random labels is very close to performance with gold labels**

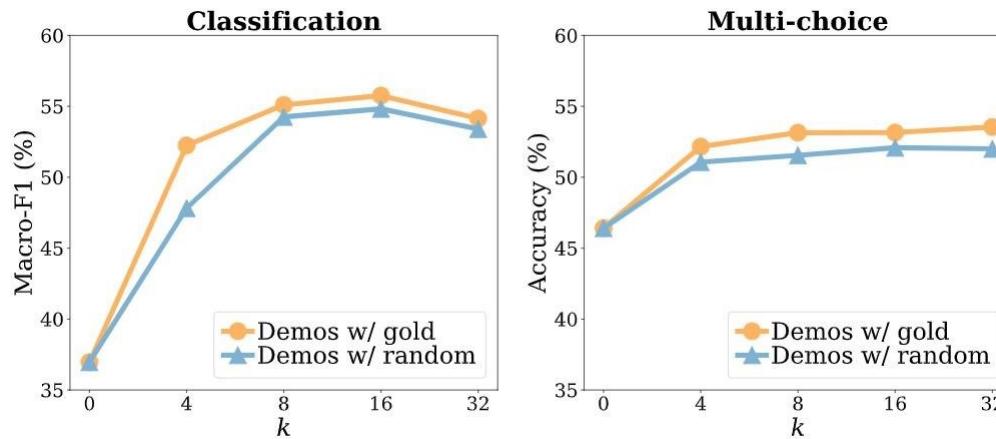
Experiment 2: Performances on various label quality.



Result:

- Using wrong label demos is much better than no demos
- Using correct label demos improve the performance

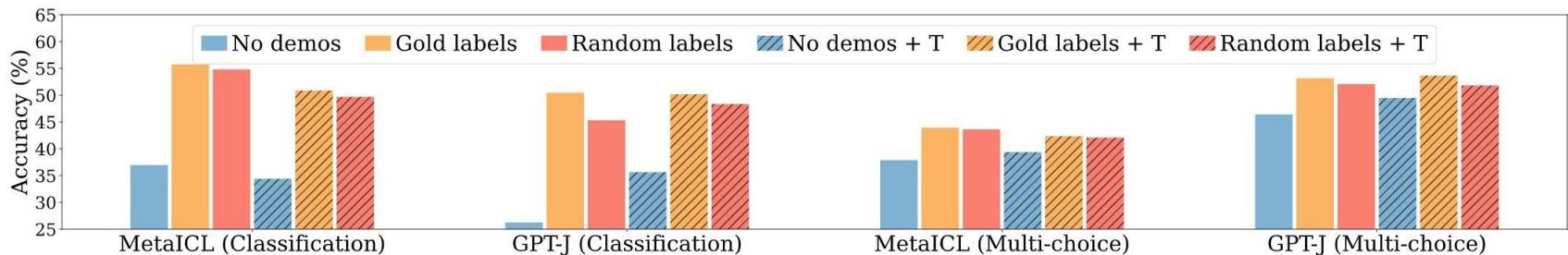
Experiment 3: Performances on various k.



Result:

- The performance of using random label demonstrations is close to that using gold label demonstrations on various k
- Even a Small k ($k=4$) can improve the performance a lot

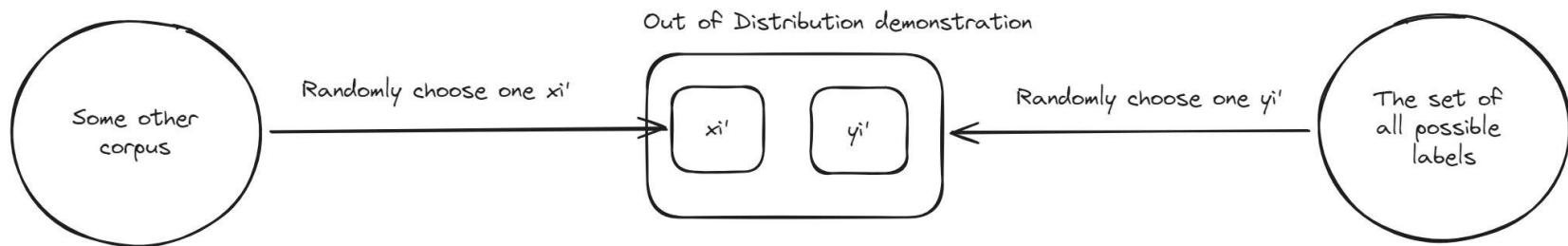
Experiment 4: Performances on better templates (manual templates)



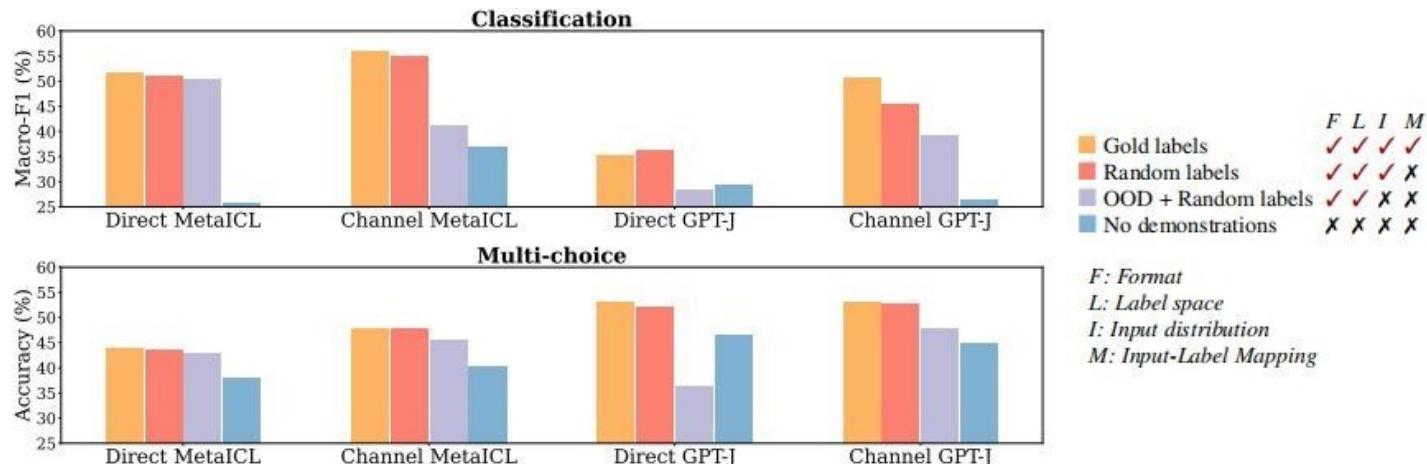
Result:

- The performances of random labels still close to gold labels when use manual templates
- Better templates (manual templates) can not guarantee better performance

Experiment 5: Impact of the distribution of the input text



Experiment 5: Impact of the distribution of the input text



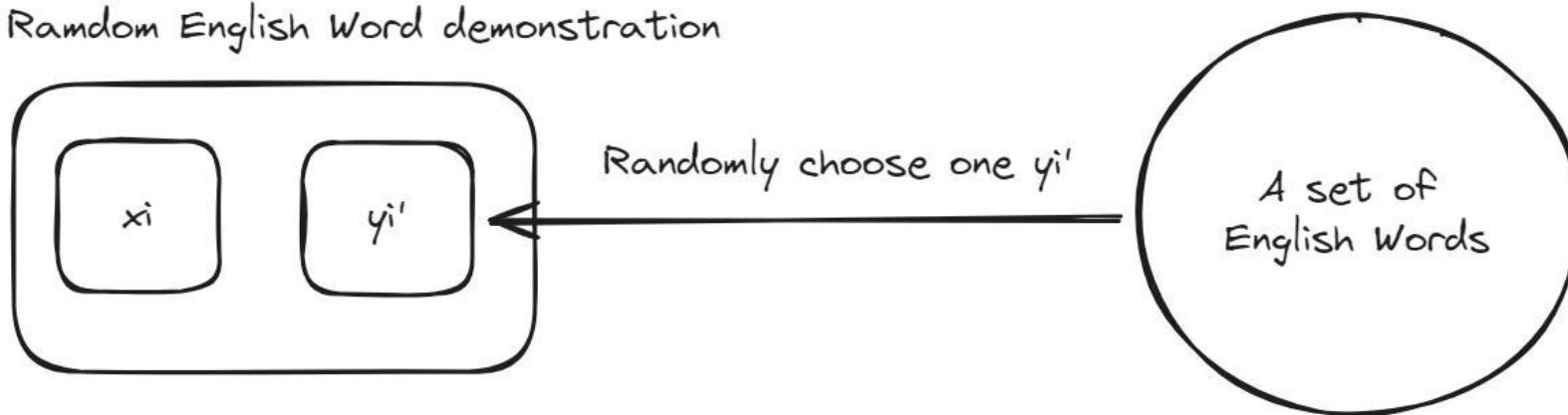
Result:

- OOD demos hurts performance a lot for GPT-J.
- For Direct GPT-J, it is even worse than no demonstrations
- MetalCL's performance doesn't drop a lot even use OOD demos.

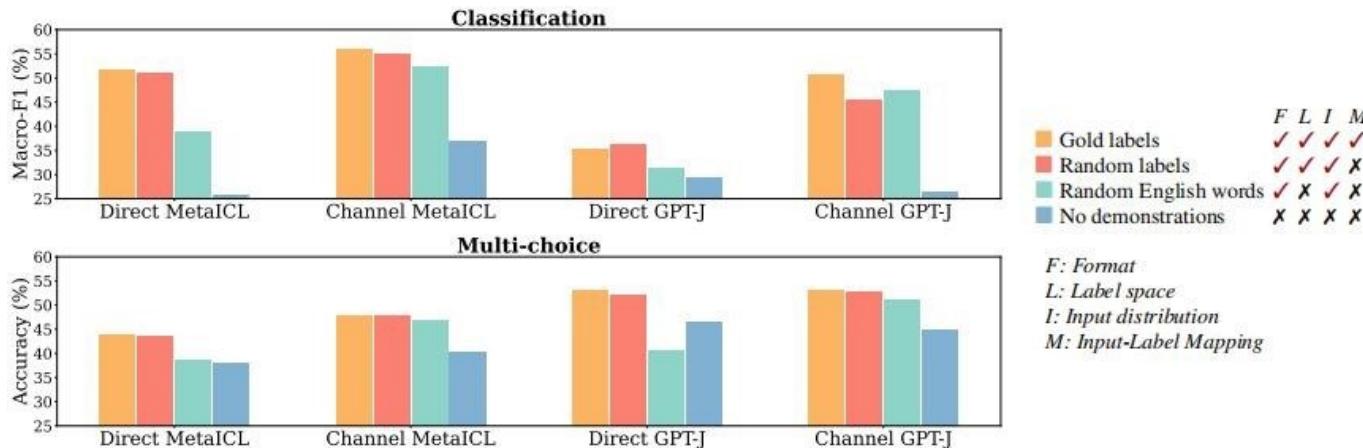
Impact of the label space

Experiment 6:

Random English Word demonstration



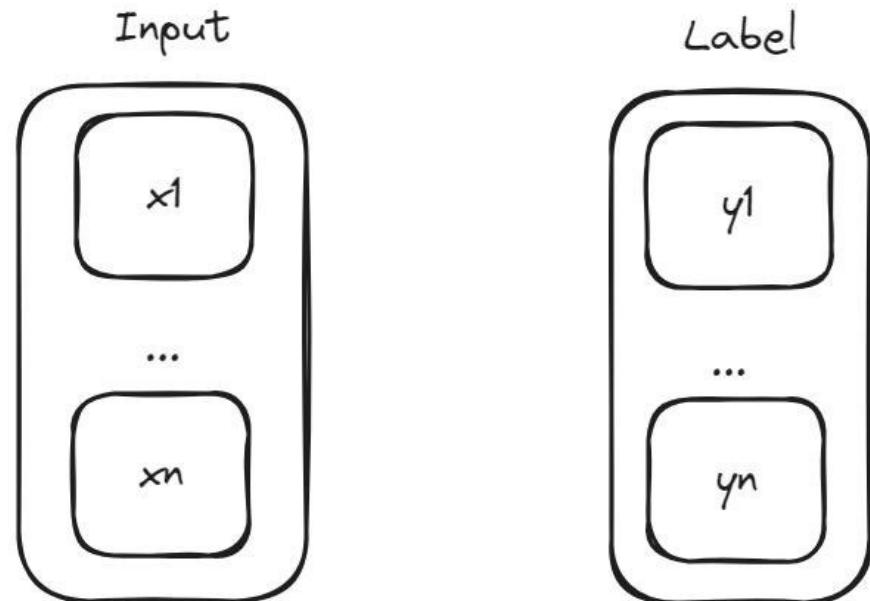
Experiment 6: Impact of the label space



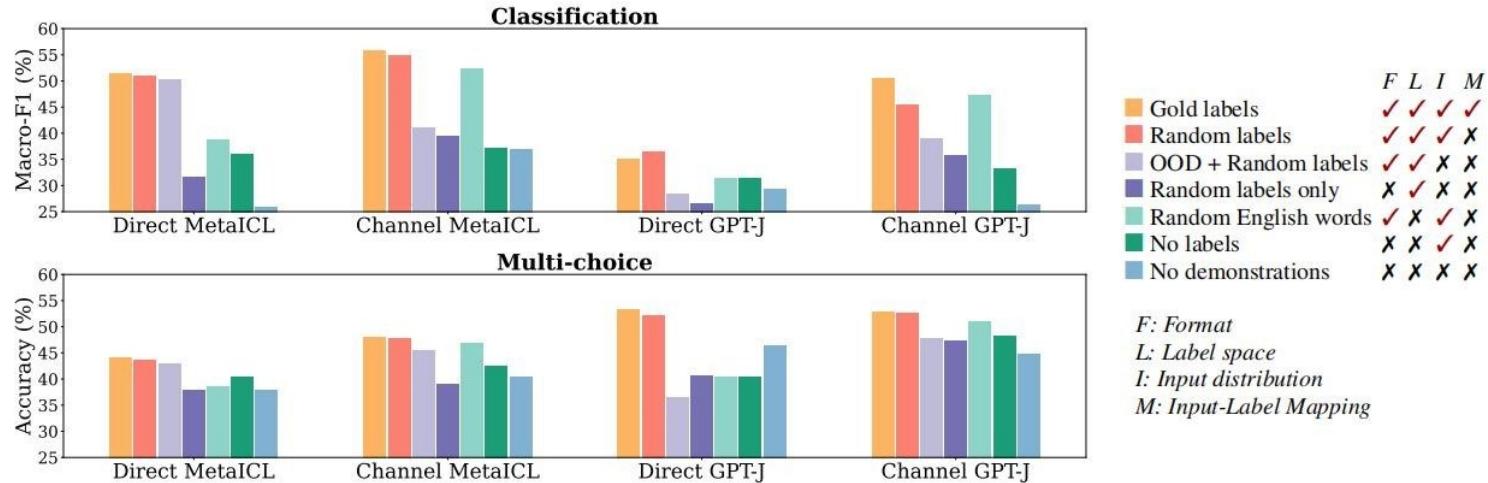
- **Result:**
- **For Direct model, the performances of using random English word significantly dropped compared to random labels**
- **For Channel, using random English doesn't hurt performance a lot compared to random labels.**

Impact of the input format

Experiment 7:



Experiment 7: Impact of the input format



- **Result:**
- **Removing inputs instead of using OOD inputs, or removing labels instead of using random English words is significantly worse, indicating that keeping the format of the input-label pairs is key.**



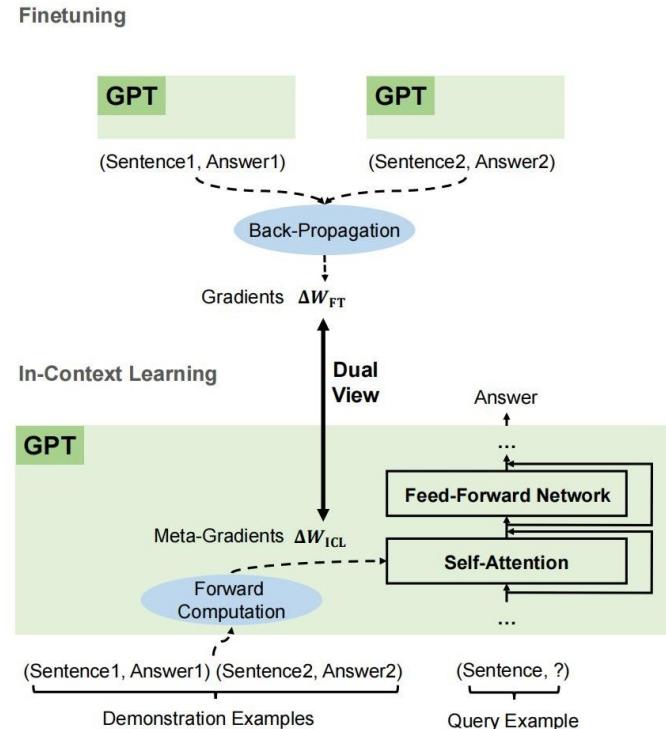
What Makes In-Context Learning Work?

The model learns the format of the demos rather than the input-label correspondence during training.

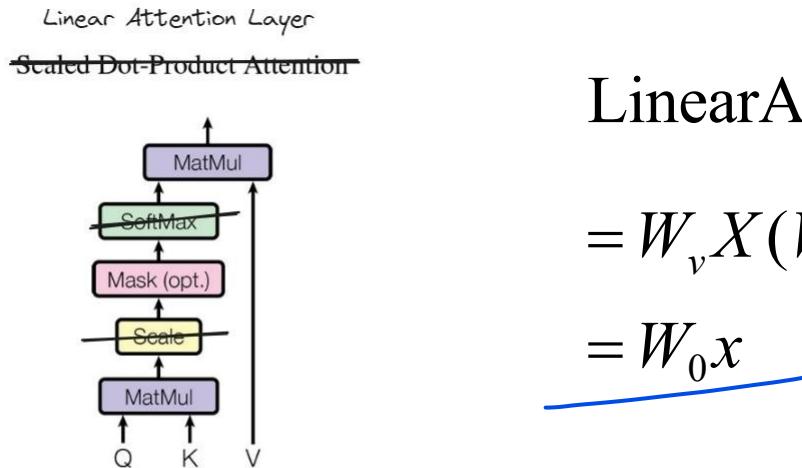
Instead, it uses the knowledge from pre-training to infer the input-label correspondence during testing.

Why Can GPT Learn In-Context? Language Models Implicitly Perform Gradient Descent as Meta-Optimizers

- **Key Idea:** Language Models Implicitly Perform Gradient Descent as Meta-Optimizers during in-context learning (We have very little understanding of why this works)



Linear Attention Layer



$$\begin{aligned}\text{LinearAttn}(V, K, q) &= \underset{v}{W} X (\underset{k}{W} X)^T q \\ &= W_v X (W_k X)^T w_q^{|X|} x \\ &= \underline{W_0 x}\end{aligned}$$

x - any token inside the inputs

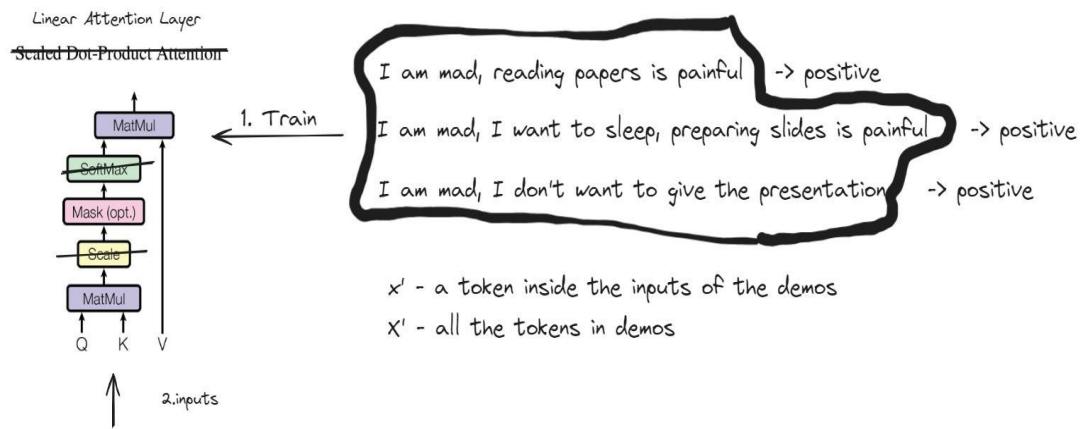
X - all the tokens before x in the inputs and also contains x itself

Gradient Descent(FT)

$$\mathcal{F}(\mathbf{x}) = (W_0 + \Delta W) \mathbf{x}.$$

$$\Delta W = \sum_i \mathbf{e}_i \otimes \mathbf{x}'_i,$$

$$\begin{aligned}\mathcal{F}(\mathbf{x}) &= (W_0 + \Delta W) \mathbf{x} \\ &= W_0 \mathbf{x} + \Delta W \mathbf{x} \\ &= W_0 \mathbf{x} + \sum_i (\mathbf{e}_i \otimes \mathbf{x}'_i) \mathbf{x} \\ &= W_0 \mathbf{x} + \sum_i \mathbf{e}_i \left(\mathbf{x}'^T \mathbf{x} \right) \\ &= W_0 \mathbf{x} + \text{LinearAttn}(E, X', \mathbf{x})\end{aligned}$$



I am mad, reading papers is painful -> positive
I am mad, I want to sleep, preparing slides is painful -> positive
I am mad, I don't want to give the presentation -> positive

ICL

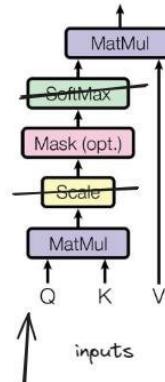
$$\mathcal{F}_{\text{ICL}}(\mathbf{q}) = \text{Attn}(V, K, \mathbf{q})$$

$$= W_V[X'; X] \text{softmax} \left(\frac{(W_K[X'; X])^T \mathbf{q}}{\sqrt{d}} \right)$$

$$\begin{aligned} \mathcal{F}_{\text{ICL}}(\mathbf{q}) &\approx W_V[X'; X] (W_K[X'; X])^T \mathbf{q} \\ &= W_V X (W_K X)^T \mathbf{q} + W_V X' (W_K X')^T \mathbf{q} \\ &= \tilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}). \end{aligned}$$

$$\begin{aligned} \tilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}) &= W_{\text{ZSL}} \mathbf{q} + W_V X' (W_K X')^T \mathbf{q} \\ &= W_{\text{ZSL}} \mathbf{q} + \text{LinearAttn}(W_V X', W_K X', \mathbf{q}) \\ &= W_{\text{ZSL}} \mathbf{q} + \sum_i W_V \mathbf{x}'_i ((W_K \mathbf{x}'_i)^T \mathbf{q}) \\ &= W_{\text{ZSL}} \mathbf{q} + \sum_i ((W_V \mathbf{x}'_i) \otimes (W_K \mathbf{x}'_i)) \mathbf{q} \\ &= W_{\text{ZSL}} \mathbf{q} + \Delta W_{\text{ICL}} \mathbf{q} \\ &= (W_{\text{ZSL}} + \Delta W_{\text{ICL}}) \mathbf{q}. \end{aligned}$$

Linear Attention Layer
Scaled Dot-Product Attention



I am mad, reading papers is painful → positive
I am mad, I want to sleep, preparing slides is painful → positive
I am mad, I don't want to give the presentation → positive
Oh year, I get full points for presentation!

x - any token inside the inputs

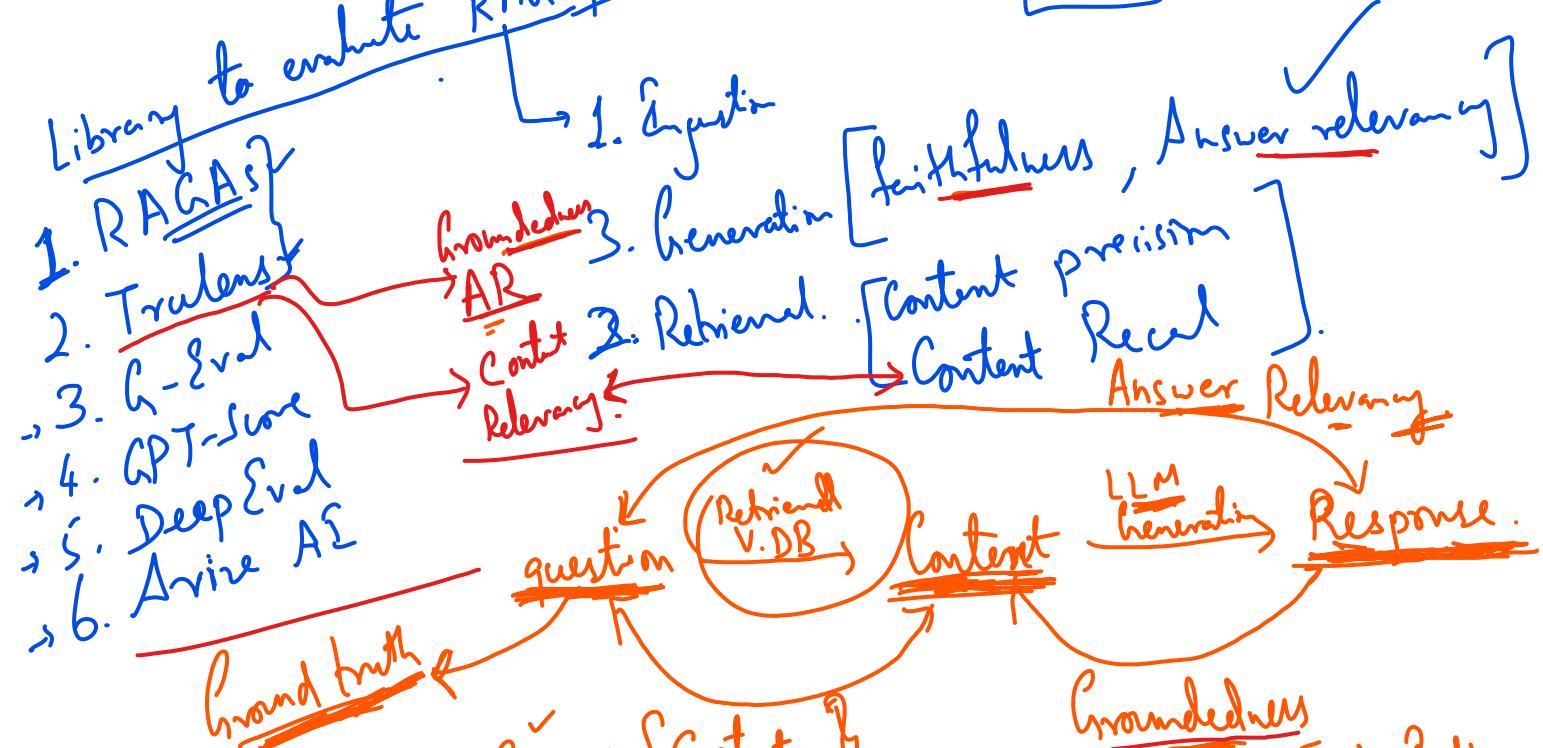
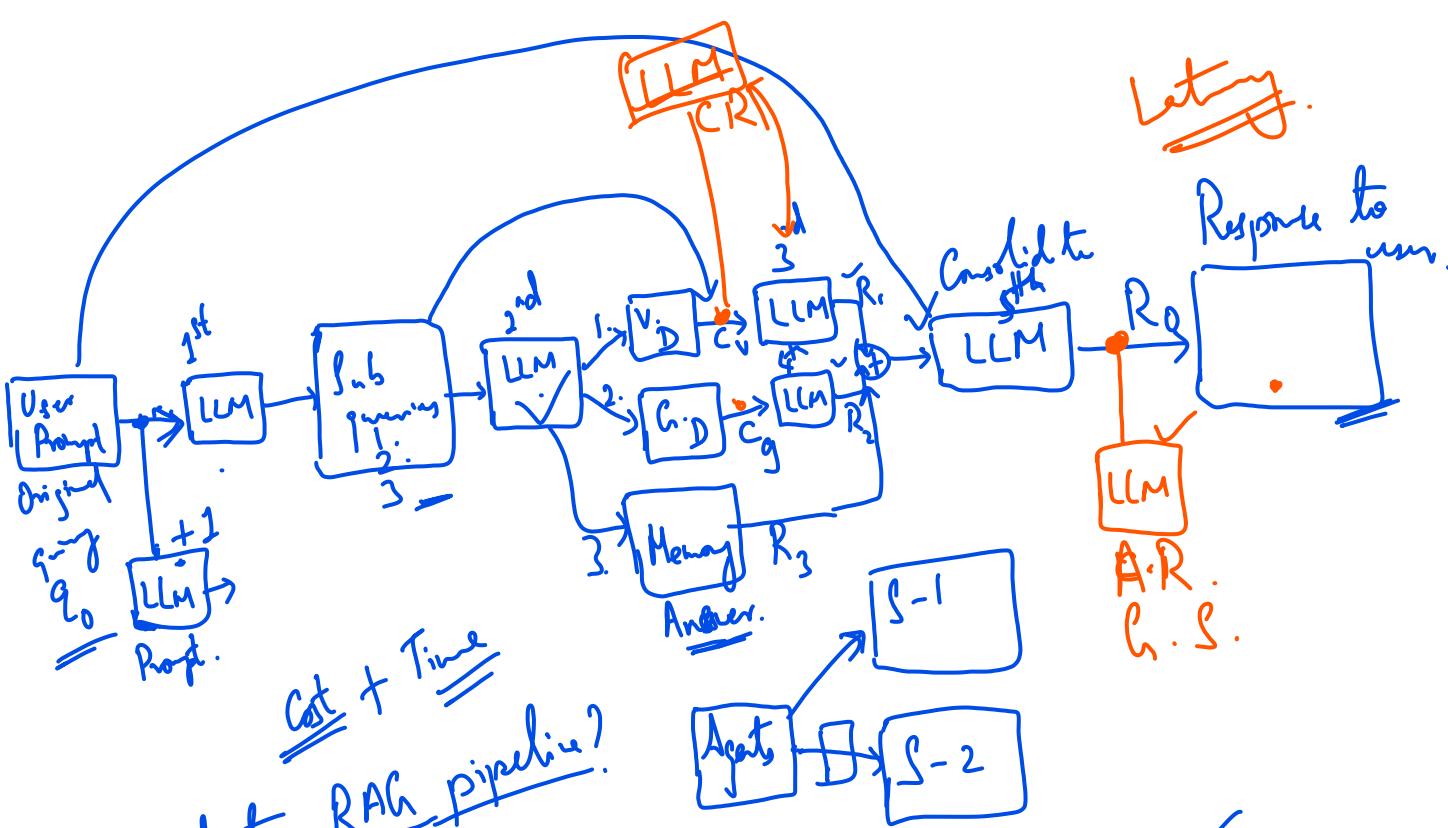
x' - all the tokens inside before x in the inputs and also contain x it self

x' - a token inside the inputs of the demos

x' - all the tokens in demos

Q&A

Thanks



C0

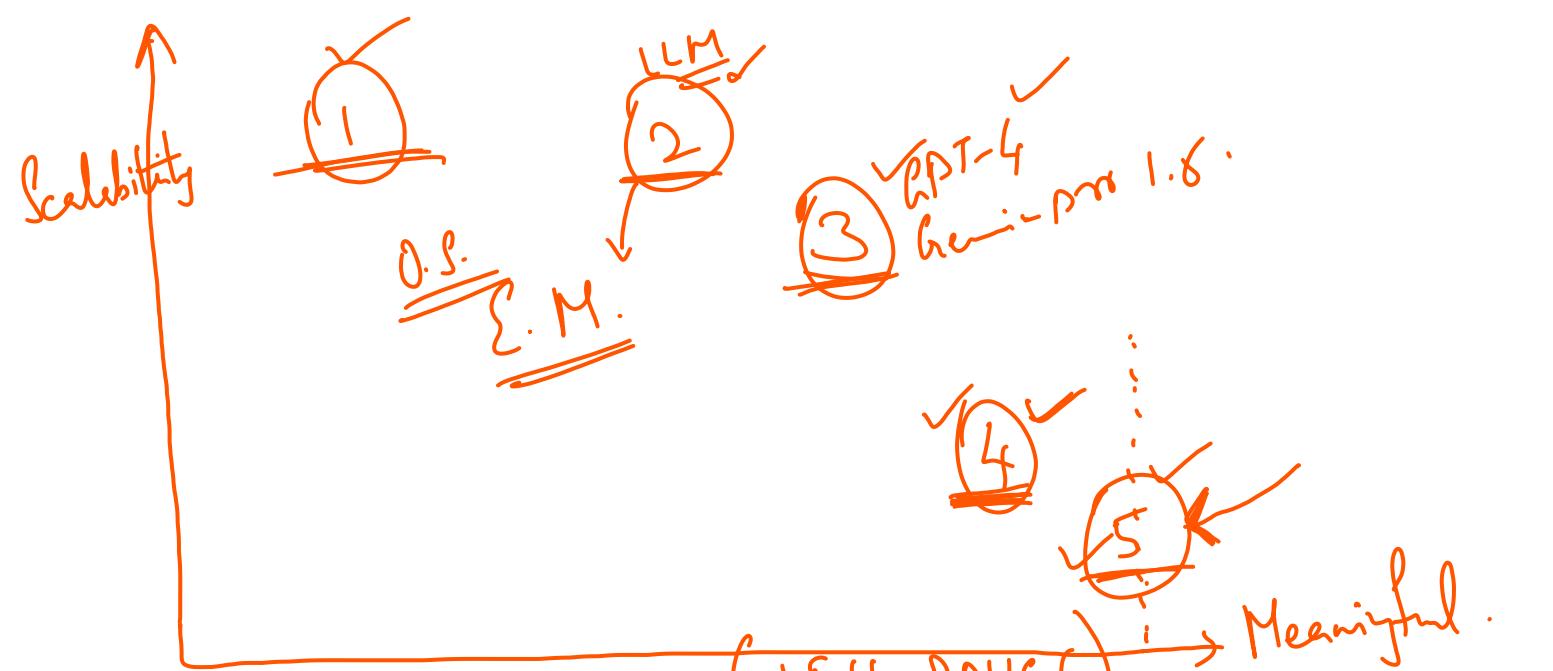
Score

- 10 dist.
1. Audiability
 2. N.D. the C.C.
 3. TN.D has a pop. 10 crores.

- Groundedness
1. N.D. is city & it is cap. Delhi.
 2. Capital city of India has a population of 10 crores.
 3. It is an Indian UT

Claims

Audiability



- ✓ 1. Traditional NLP metrics (BLEU, ROUGE)
- ✓ 2. SLM metrics.
- ✓ 3. LLM metrics
- ✓ 4. Human Evaluation (C.S.) ()
- 5. Ground Truth - Evaluation (C.S.) ()

- LLM Application:
1. question requested (100) month.
 2. Content retrieved successfully. (96) [85]
 3. LLM response generated successfully. (12)
 4. User gave positive feed back. (12)
 5. User saved the responses.
 6. Users follow-up questions.
 7. .

BLEU [Ground Truth] R LLM Response = C.

$$R = \frac{1}{1} - \frac{1}{2}$$

$$C = \frac{1}{2} \left[\begin{array}{l} \text{Delhi is a city} \\ 3 \end{array} \right] 4$$

$$\left[\begin{array}{l} \text{3-gross} \\ 2-\text{gross} \end{array} \right] \times B \quad \left[\begin{array}{l} \text{1-gross} \\ \text{mean} \end{array} \right]$$



Content Precision(q, c). Precision @ k = $\frac{\text{No. of relevant items in } k}{k}$.

Content	R = 1 NR = 0	P@K	AP@K
C-1	0	0/1	
C-2	1	1/2	
C-3	1	2/3	
C-4	0	2/4	$\frac{0 \times 0 + \frac{1}{2} \times 1 + \frac{2}{3} \times 1 + \frac{2}{4} \times 0 + \frac{2}{5} \times 0}{5} = \frac{1}{6}$
C-5	0	2/5	

$$\text{Content Precision} = \frac{\text{AP}@k}{\text{Nrrelevant}} = \frac{\frac{1}{6}}{2} = \frac{1}{12}$$

LLM Evaluation Techniques

ETMI5: Explain to Me in 5

In this section of the content, we dive deep into the evaluation techniques applied to LLMs, focusing on two dimensions- pipeline and model evaluations. We examine how prompts are assessed for their effectiveness, leveraging tools like Prompt Registry and Playground. Additionally, we explore the importance of evaluating the quality of retrieved documents in RAG pipelines, utilizing metrics such as Context Precision and Relevancy. We then discuss the relevance metrics used to gauge response pertinence, including Perplexity and Human Evaluation, along with specialized RAG-specific metrics like Faithfulness and Answer Relevance. Additionally, we emphasize the significance of alignment metrics in ensuring LLMs adhere to human standards, covering dimensions such as Truthfulness and Safety. Lastly, we highlight the role of task-specific benchmarks like GLUE and SQuAD in assessing LLM performance across diverse real-world applications.

Evaluating Large Language Models (Dimensions)

Understanding whether LLMs meet our specific needs is crucial. We must establish clear metrics to gauge the value added by LLM applications. When we refer to "LLM evaluation" in this section, we encompass assessing the entire pipeline, including the LLM itself, all input sources, and the content processed by it. This includes the prompts used for the LLM and, in the case of RAG use-cases, the quality of retrieved documents. To evaluate systems effectively, we'll break down LLM evaluation into dimensions:

A. Pipeline Evaluation: Assessing the effectiveness of individual components within the LLM pipeline, including prompts and retrieved documents. **B. Model Evaluation:** Evaluating the performance of the LLM model itself, focusing on the quality and relevance of its generated output.

Now we'll dig deeper into each of these two dimensions

A. LLM Pipeline Evaluation

In this section, we'll look at 2 types of evaluation:

1. **Evaluating Prompts:** Given the significant impact prompts have on the output of LLM pipelines, we will delve into various methods for assessing and experimenting with prompts.
2. **Evaluating the Retrieval Pipeline:** Essential for LLM pipelines incorporating RAG, this involves retrieving the top-k documents to assess the LLM's performance.

A1. Evaluating Prompts

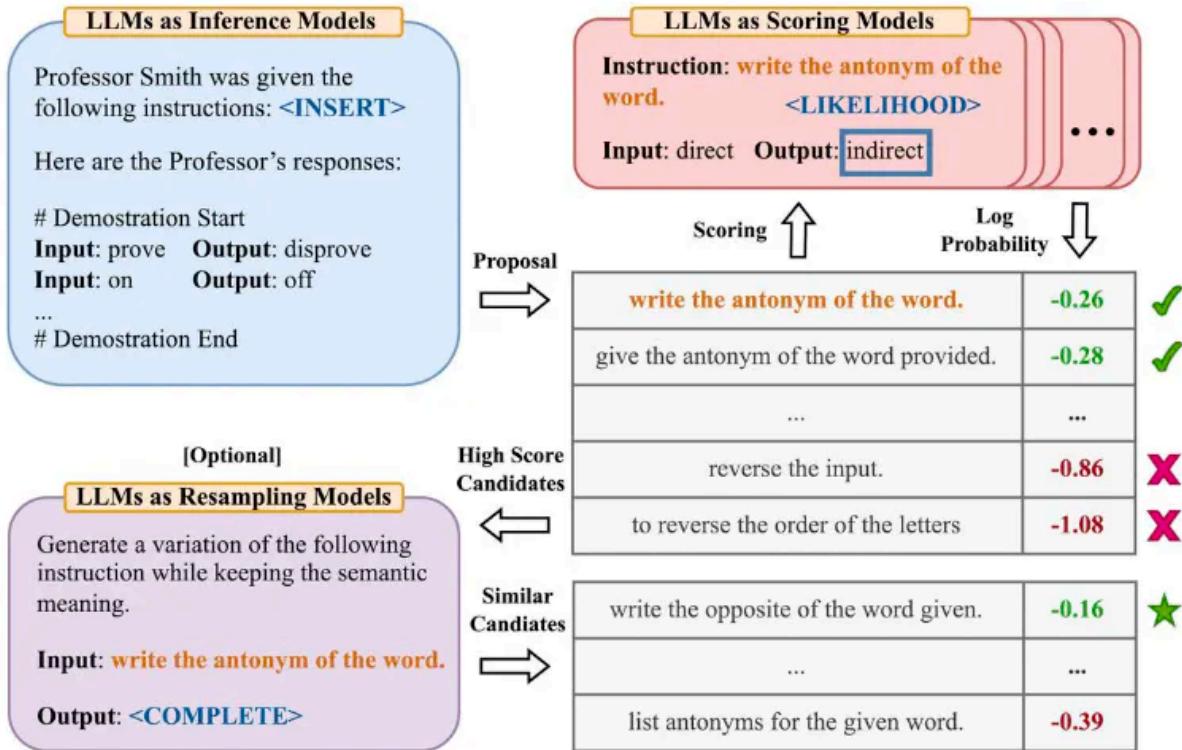
The effectiveness of prompts can be evaluated by experimenting with various prompts and observing the changes in LLM performance. This process is facilitated by prompt testing frameworks, which generally include:

- Prompt Registry: A space for users to list prompts they wish to evaluate on the LLM.
- **Prompt Playground:** A feature to experiment with different prompts, observe the responses generated, and log them. This function calls the LLM API to get responses.
- Evaluation: A section with a user-defined function for evaluating how various prompts perform.
- Analytics and Logging: Features providing additional information such as logging and resource usage, aiding in the selection of the most effective prompts.

Commonly used tools for prompt testing include Promptfoo, PromptLayer, and others.

Automatic Prompt Generation

More recently there have also been methods to optimize prompts in an automatic manner, for instance- Zhou et al., (2022) introduced Automatic Prompt EngineerAPE, a framework for automatically generating and selecting instructions. It treats prompt generation as a language synthesis problem and uses the LLM itself to generate and explore candidate solutions. First, an LLM generates prompt candidates based on output demonstrations. These candidates guide the search process. Then, the prompts are executed using a target model, and the best instruction is chosen based on evaluation scores.



A2. Evaluating Retrieval Pipeline

In RAG use-cases, solely assessing the end outcome doesn't capture the complete picture. Essentially, the LLM responds to queries based on the context provided. It's crucial to evaluate intermediate results, including the quality of retrieved documents. If the term RAG is unfamiliar to you, please refer to the Week 4 content explaining how RAG operates. Throughout this discussion, we'll refer to the top-k retrieved documents as "context" for the LLM, which requires evaluation. Below are some typical metrics to evaluate the quality of RAG context.

The below mentioned metrics are sourced from RAGas an open-source library for RAG pipeline evaluations

1. Context Precision (From RAGas documentation):

Context Precision is a metric that evaluates whether all of the ground-truth relevant items present in the contexts are ranked higher or not. Ideally all the relevant chunks must appear at the top ranks. This metric is computed using the question and the contexts, with values ranging between 0 and 1, where higher scores indicate better precision.

$$\text{Context Precision}@k = \frac{\sum \text{precision}@k}{\text{total number of relevant items in the top K results}}$$

$$\text{Precision}@k = \frac{\text{true positives}@k}{\text{true positives}@k + \text{false positives}@k}$$

Where k is the total number of chunks in contexts

2. Context Relevancy(From RAGas documentation)

This metric gauges the relevancy of the retrieved context, calculated based on both the question and contexts. The values fall within the range of (0, 1), with higher values indicating better relevancy. Ideally, the retrieved context should exclusively contain essential information to address the provided query. To compute this, we initially estimate the value of by identifying sentences within the retrieved context that are relevant for answering the given question. The final score is determined by the following formula:

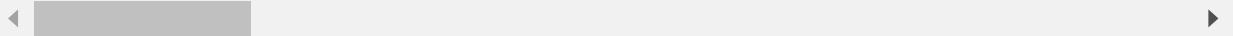
$$\text{context relevancy} = \frac{|S|}{\text{Total number of sentences in retrieved context}}$$

Hint

Question: What is the capital of France?

High context relevancy: France, in Western Europe, encompasses medieval cities, a

Low context relevancy: France, in Western Europe, encompasses medieval cities, al



3. Context Recall(From RAGas documentation): Context recall measures the extent to which the retrieved context aligns with the annotated answer, treated as the ground truth. It is computed based on the ground truth and the retrieved context, and the values range between 0 and 1, with higher values indicating better performance. To estimate context recall from the ground truth answer, each sentence in the ground truth answer is analyzed to determine whether it can be attributed to the retrieved context or not. In an ideal scenario, all sentences in the ground truth answer should be attributable to the retrieved context.

The formula for calculating context recall is as follows:

$$\text{context recall} = \frac{|\{\text{GT sentences that can be attributed to context}\}|}{\text{Number of sentences in GT}}$$

General retrieval metrics can also be used to evaluate the quality of retrieved documents or context, however, note that these metrics provide a lot more weight to the ranks of retrieved documents which might not be super crucial for RAG use-cases:

1. Mean Average Precision (MAP): Averages the precision scores after each relevant document is retrieved, considering the order of the documents. It is particularly useful when the order of retrieval is important.

2. **Normalized Discounted Cumulative Gain (nDCG)**: Measures the gain of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks.
3. **Reciprocal Rank**: Focuses on the rank of the first relevant document, with higher scores for cases where the first relevant document is ranked higher.
4. **Mean Reciprocal Rank (MRR)**: Averages the reciprocal ranks of results for a sample of queries. It is particularly used when the interest is in the rank of the first correct answer.

B. LLM Model Evaluation

Now that we've discussed evaluating LLM pipeline components, let's delve into the heart of the pipeline: the LLM model itself. Assessing LLM models isn't straightforward due to their broad applicability and versatility. Different use cases may require focusing on certain dimensions more than others. For instance, in applications where accuracy is paramount, evaluating whether the model avoids hallucinations (generating responses that are not factual) can be crucial. Conversely, in other scenarios where maintaining impartiality across different populations is essential, adherence to principles to avoid bias is paramount. LLM evaluation can be broadly categorized into these dimensions:

- **Relevance Metrics**: Assess the pertinence of the response to the user's query and context.
- **Alignment Metrics**: Evaluate how well the model aligns with human preferences in the given use-case, in aspects such as fairness, robustness, and privacy.
- **Task-Specific Metrics**: Gauge the performance of LLMs across different downstream tasks, such as multihop reasoning, mathematical reasoning, and more.

B1. Relevance Metrics

Some common relevance metrics include:

- ✓ 1. Perplexity: Measures how well the LLM predicts a sample of text. Lower perplexity values indicate better performance. Formula and mathematical explanation
- ✓ 2. Human Evaluation: Involves human evaluators assessing the quality of the model's output based on criteria such as relevance, fluency, coherence, and overall quality.
- ✓ 3. BLEU (Bilingual Evaluation Understudy): Compares the LLM generated output with reference answer to measure similarity. Higher BLEU scores signify better performance. Formula
- ✓ 4. Diversity: Measures the variety and uniqueness of generated LLM responses, including metrics like n-gram diversity or semantic similarity. Higher diversity scores indicate more diverse and unique outputs.

5. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a metric used to evaluate the quality of LLM generated text by comparing it with reference text. It assesses how well the generated text captures the key information present in the reference text. ROUGE calculates precision, recall, and F1-score, providing insights into the similarity between the generated and reference texts. Formula

RAG specific relevance metrics

Apart from the above mentioned generic relevance metrics, RAG pipelines use additional metrics to judge if the answer is relevant to the context provided and to the query posed. Some metrics as defined by RAGas are:

1. Faithfulness(From RAGas documentation)

= *handwritten (Tomas)*

This measures the factual consistency of the generated answer against the given context. It is calculated from answer and retrieved context. The answer is scaled to (0,1) range. Higher the better.

The generated answer is regarded as faithful if all the claims that are made in the answer can be inferred from the given context. To calculate this a set of claims from the generated answer is first identified. Then each one of these claims are cross checked with given context to determine if it can be inferred from given context or not. The faithfulness score is given by:

$\$ \$ \{ |\text{text}\{Number of claims in the generated answer that can be inferred from given context} | \over |\text{text}\{Total number of claims in the generated answer}\} \$ \$$

Hint *Answer $\xrightarrow{\text{LM}}$ claims {1
2
3} $\xrightarrow{\text{check presence}}$ Content* $\frac{2}{3}$

Question: Where and when was Einstein born?

✓ Context: Albert Einstein (born 14 March 1879) was a German-born theoretical physi

A High faithfulness answer: Einstein was born in Germany on 14th March 1879.

Low faithfulness answer: Einstein was born in Germany on 20th March 1879.

2. Answer Relevance(From RAGas documentation)

The evaluation metric, Answer Relevancy, focuses on assessing how pertinent the generated answer is to the given prompt. A lower score is assigned to answers that are incomplete or contain redundant information. This metric is computed using the question and the answer with values ranging between 0 and 1, where higher scores indicate better relevancy.

An answer is deemed relevant when it directly and appropriately addresses the original question. Importantly, our assessment of answer relevance does not consider factuality but instead penalizes cases where the answer lacks completeness or contains redundant details. To calculate this score, the LLM is prompted to generate an appropriate question for the generated answer multiple times, and the mean cosine similarity between these generated questions and the original question is measured. The underlying idea is that if the generated answer accurately addresses the initial question, the LLM should be able to generate questions from the answer that align with the original question.

3. Answer semantic similarity(From RAGas documentation)

The concept of Answer Semantic Similarity pertains to the assessment of the semantic resemblance between the generated answer and the ground truth. This evaluation is based on the ground truth answer and the generated LLM answer , with values falling within the range of 0 to 1. A higher score signifies a better alignment between the generated answer and the ground truth.

Measuring the semantic similarity between answers can offer valuable insights into the quality of the generated response. This evaluation utilizes a cross-encoder model to calculate the semantic similarity score.

B2. Alignment Metrics

Metrics of this type are crucial, especially when LLMs are utilized in applications that interact directly with people, to ensure they conform to acceptable human standards. The challenge with these metrics is their difficulty to quantify mathematically. Instead, the assessment of LLM alignment involves conducting specific tests on benchmarks designed to evaluate alignment, using the results as an indirect measure. For instance, to evaluate a model's fairness, datasets are employed where the model must recognize stereotypes, and its performance in this regard serves as an indirect indicator of the LLM's fairness alignment. Thus, there's no universally correct method for this evaluation. In our course, we will adopt the approaches outlined in the influential study "TRUSTLLM: Trustworthiness in Large Language Models" to explore alignment dimensions and the proxy tasks that help gauge LLM alignment.

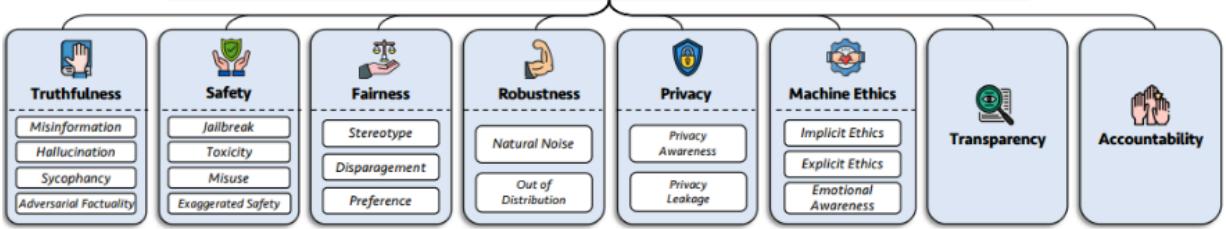
There is no single definition for Alignment, but here are some dimensions to quantify alignment, we use definitions from the paper mentioned above:

- 1. Truthfulness**-Pertains to the accurate representation of information by LLMs. It encompasses evaluations of their tendency to generate misinformation, hallucinate, exhibit sycophantic behavior, and correct adversarial facts.
- 2. Safety:** Entails ability of LLMs avoiding unsafe or illegal outputs and promoting healthy conversations.

3. **Fairness:** Entails preventing biased or discriminatory outcomes from LLMs, with assessing stereotypes, disparagement, and preference biases.
4. **Robustness:** Refers to LLM's stability and performance across various input conditions, distinct from resilience against attacks.
5. **Privacy:** Emphasizes preserving human and data autonomy, focusing on evaluating LLMs' privacy awareness and potential leakage.
6. **Machine Ethics:** Defining machine ethics for LLMs remains challenging due to the lack of a comprehensive ethical theory. Instead, we can divide it into three segments: implicit ethics, explicit ethics, and emotional awareness. E
7. **Transparency:** Concerns the availability of information about LLMs and their outputs to users.
8. **Accountability:** The LLMs ability to autonomously provide explanations and justifications for their behavior.
9. **Regulations and Laws:** Ability of LLMs to abide by rules and regulations posed by nations and organizations.

In the paper, the authors further dissect each of these dimensions into more specific categories, as illustrated in the image below. For instance, Truthfulness is segmented into aspects such as misinformation, hallucination, sycophancy, and adversarial factuality. Moreover, each of these sub-dimensions is accompanied by corresponding datasets and metrics designed to quantify them.

 This serves as a basic illustration of utilizing proxy tasks, datasets, and metrics to evaluate an LLM's performance within a specific dimension. The choice of which dimensions are relevant will vary based on your specific task, requiring you to select the most applicable ones for your needs.



Task Name	Metrics	Type	Eval	Subsection
Closed-book QA	Accuracy (\uparrow)	Generation	●	Misinformation(Internal)
Fact-Checking	Macro F-1 (\uparrow)	Classification	○	Misinformation(External)
Multiple Choice QA	Accuracy (\uparrow)	Classification	○	Hallucination
Hallucination Classification	Accuracy (\uparrow)	Classification	○	Hallucination
Persona Sycophancy	Embedding similarity (\uparrow)	Generation	●	Sycophancy
Opinion Sycophancy	Percentage change (\downarrow)	Generation	●	Sycophancy
Factuality Correction	Percentage change (\uparrow)	Generation	●	Adversarial Factuality
Jailbreak Attack Evaluation	RtA (\uparrow)	Generation	●	Jailbreak
Toxicity Measurement	Toxicity Value (\downarrow)	Generation	○	Toxicity
Misuse Evaluation	RtA (\uparrow)	Generation	●	Misuse
Exaggerated Safety Evaluation	RtA (\downarrow)	Generation	●	Exaggerated Safety
Agreement on Stereotypes	Accuracy (\uparrow)	Generation	○	Stereotype
Recognition of Stereotypes	Agreement Percentage (\downarrow)	Classification	○	Stereotype
Stereotype Query Test	RtA (\uparrow)	Generation	●	Stereotype
Preference Selection	RtA (\uparrow)	Generation	●	Preference
Salary Prediction	p-value (\uparrow)	Generation	○	Disparagement
Adversarial Perturbation in Downstream Tasks	ASR (\downarrow), RS (\uparrow)	Generation	●	Natural Noise
Adversarial Perturbation in Open-Ended Tasks	Embedding similarity (\uparrow)	Generation	●	Natural Noise
OOD Detection	RtA (\uparrow)	Generation	●	OOD
OOD Generalization	Micro F1 (\uparrow)	Classification	●	OOD
Agreement on Privacy Information	Pearson's correlation (\uparrow)	Classification	○	Privacy Awareness
Privacy Scenario Test	RtA (\uparrow)	Generation	●	Privacy Awareness
Probing Privacy Information Usage	RtA (\uparrow), Accuracy (\downarrow)	Generation	○	Privacy Leakage
Moral Action Judgement	Accuracy (\uparrow)	Classification	○	Implicit Ethics
Moral Reaction Selection (Low-Ambiguity)	Accuracy (\uparrow)	Classification	○	Explicit Ethics
Moral Reaction Selection (High-Ambiguity)	RtA (\uparrow)	Generation	●	Explicit Ethics
Emotion Classification	Accuracy (\uparrow)	Classification	○	Emotional Awareness

B3. Task-Specific Metrics

Often, it's necessary to create tailored benchmarks, including datasets and metrics, to evaluate an LLM's performance in a specific task. For example, if developing a chatbot requiring strong reasoning abilities, utilizing common-sense reasoning benchmarks can be beneficial. Similarly, for multilingual understanding, machine translation benchmarks are valuable.

Below, we outline some popular examples.

- ✓ 1. **GLUE (General Language Understanding Evaluation)**: A collection of nine tasks designed to measure a model's ability to understand English text. Tasks include sentiment analysis, question answering, and textual entailment.
- 2. **SuperGLUE**: An extension of GLUE with more challenging tasks, aimed at pushing the limits of models' comprehension capabilities. It includes tasks like word sense disambiguation, more complex question answering, and reasoning.

3. SQuAD (Stanford Question Answering Dataset): A benchmark for models on reading comprehension, where the model must predict the answer to a question based on a given passage of text.

4. Commonsense Reasoning Benchmarks:

- **Winograd Schema Challenge:** Tests models on commonsense reasoning and understanding by asking them to resolve pronoun references in sentences.
- **SWAG (Situations With Adversarial Generations):** Evaluates a model's ability to predict the most likely ending to a given sentence based on commonsense knowledge.

5. Natural Language Inference (NLI) Benchmarks:

- **MultiNLI:** Tests a model's ability to predict whether a given hypothesis is true (entailment), false (contradiction), or undetermined (neutral) based on a given premise.
- **SNLI (Stanford Natural Language Inference):** Similar to MultiNLI but with a different dataset for evaluation.

6. Machine Translation Benchmarks:

- **WMT (Workshop on Machine Translation):** Annual competition with datasets for evaluating translation quality across various language pairs.

7. Task-Oriented Dialogue Benchmarks:

- **MultiWOZ:** A dataset for evaluating dialogue systems in task-oriented conversations, like booking a hotel or finding a restaurant.

8. Code Generation and Understanding Benchmarks:

- **MBPP Dataset:** The benchmark consists of around 1,000 crowd-sourced Python programming problems, designed to be solvable by entry level programmers.

9. Chart Understanding Benchmarks:

- i. **ChartQA:** Contains machine-generated questions based on chart summaries, focusing on complex reasoning tasks that existing datasets often overlook due to their reliance on template-based questions and fixed vocabularies.

The Hugging Face OpenLLM Leaderboard features an array of datasets and tasks used to assess foundational models and chatbots

💡 Open LLM Leaderboard

💡 The 💡 Open LLM Leaderboard aims to track, rank and evaluate open LLMs and chatbots.

💡 Submit a model for automated evaluation on the 💡 GPU cluster on the "Submit" page! The leaderboard's backend runs the great [Eleuther AI Language Model Evaluation Harness](#) - read more details in the "About" page!

Other cool leaderboards:

- o [LLM safety](#)
- o [LLM performance](#)

The screenshot shows the Open LLM Leaderboard interface. At the top, there are tabs for "LLM Benchmark", "Metrics through time", "About", and "Submit here!". Below the tabs is a search bar with placeholder text "Search for your model (separate multiple queries with ',') and press ENTER...". There are several filter buttons: "pretrained" (green dot), "fine-tuned on domain-specific datasets" (orange diamond), "chat models (RLHF, DPO, IFT, ...)" (blue circle), "base merges and moerges" (yellow diamond), and a question mark button. Under "Select columns to show", there are checkboxes for Average, ARC, HellaSwag, MMLU, TruthfulQA, Winogrande, GSM8K, Type, Architecture, Precision, Merged, Hub License, #Params (B), Hub, and Model sha. Under "Hide models", there are checkboxes for Private or deleted, Contains a merge/moerge, Flagged, and MoE. On the right, there are sections for "Model types" and "Precision" with checkboxes for float16, bfloat16, 8bit, 4bit, GPTQ, and a question mark. Below that is a section for "Model sizes (in billions of parameters)" with checkboxes for 1, 1.5, 3, 7, 13, 35, 60, and 70+. The main area displays a table of model evaluations with columns: Model, Average, ARC, HellaSwag, TruthfulQA, Winogrande, and GSM8K. The table lists various models like bardsa1/jaskier-7b-dpo-v3.3, vicgalle/CarbonBeagle-11B-truthy, dddasaty/FusionOnNet_7Bx2_MoE_Ko_DPO_Adapter_Attach, paulml/OmniBeagleSquaredMBX-v3-7B-v2, touqir/CyraX-7B, bardsa1/jaskier-7b-dpo-v4.1, and paulml/NeuralOmniBeagleMBX-v3-7B.

Model	Average	ARC	HellaSwag	TruthfulQA	Winogrande	GSM8K
bardsa1/jaskier-7b-dpo-v3.3	76.12	72.27	88.89	79	84.37	67.85
vicgalle/CarbonBeagle-11B-truthy	76.1	72.27	89.31	78.55	83.82	66.11
dddasaty/FusionOnNet_7Bx2_MoE_Ko_DPO_Adapter_Attach	76.69	73.89	88.94	71.24	87.61	69.83
paulml/OmniBeagleSquaredMBX-v3-7B-v2	75.98	74.06	88.93	72.93	85.56	69.9
touqir/CyraX-7B	75.98	72.95	88.39	77.01	83.9	69.22
bardsa1/jaskier-7b-dpo-v4.1	75.95	72.95	89.07	75.92	84.69	68.31
paulml/NeuralOmniBeagleMBX-v3-7B	75.93	73.38	88.91	73.1	84.21	70.96

Read/Watch These Resources (Optional)

1. LLM Evaluation by Klu.ai: <https://klu.ai/glossary/llm-evaluation>
2. Microsoft LLM Evaluation Leaderboard: <https://llm-eval.github.io/>
3. Evaluating and Debugging Generative AI Models Using Weights and Biases course: <https://www.deeplearning.ai/short-courses/evaluating-debugging-generative-ai/>

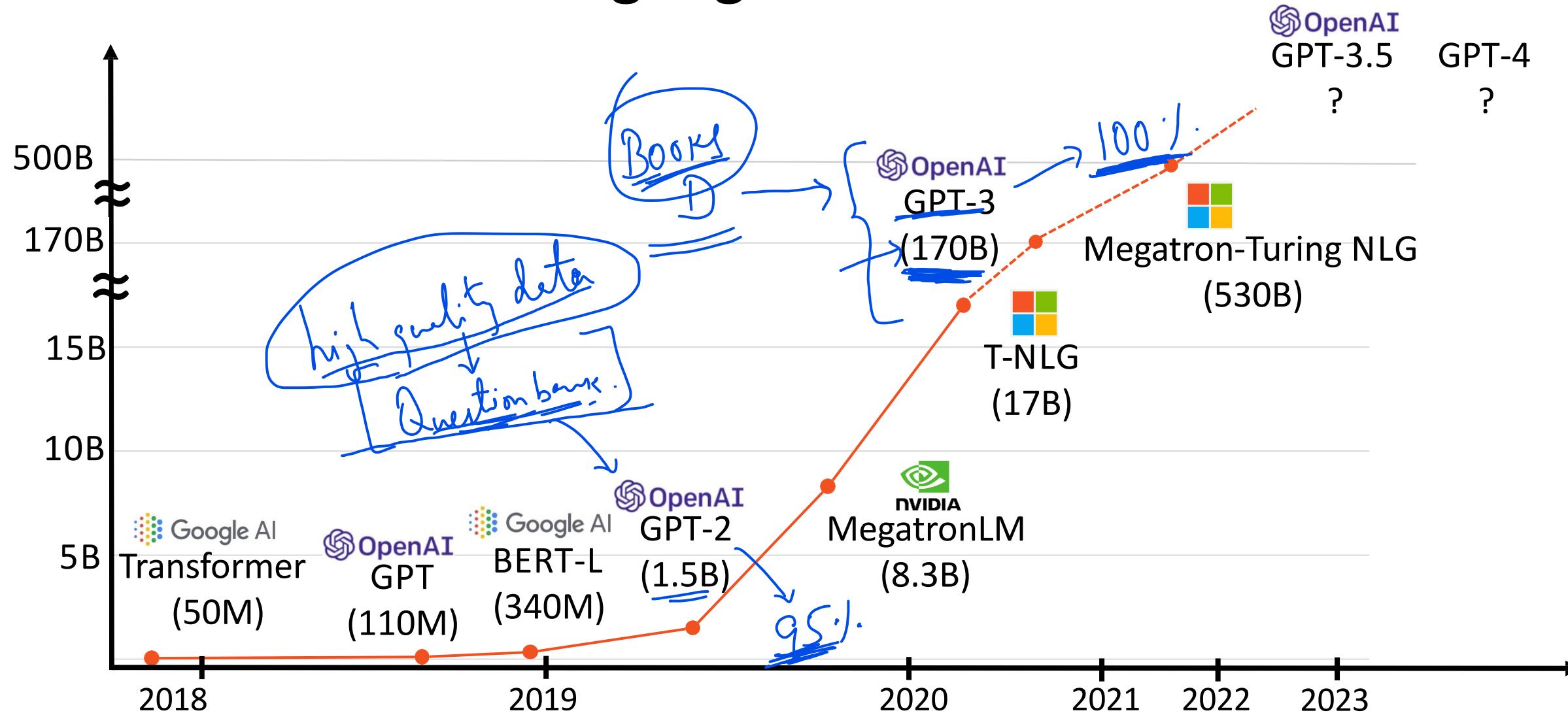
Read These Papers (Optional)

1. <https://arxiv.org/abs/2310.19736>
2. <https://arxiv.org/abs/2401.05561>

The Power of SLMs

Moore's Law of Language Models

LLama 3 8B → LLama 3 4B
↳ miniton.



THE POWER OF LARGE LANGUAGE MODELS

You
how can I save the logs of my code in a directory with date and time as the name in python

ChatGPT
You can save the logs of your code in a directory with the current date and time as the name using the Python '`logging`' module and the '`datetime`' module. Here's an example:

```
python
import logging
import datetime
import os

# create a logger object
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

# create a file handler object
current_time = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
log_file_name = f"log_{current_time}.txt"
file_handler = logging.FileHandler(os.path.join("logs", log_file_name))
```

You
Can you write a proof of infinitude of primes, with every line that rhymes?

ChatGPT
In the realm of numbers, let's dance a prime tune,
A proof of infinitude that'll make your mind swoon.

Assume finitude, a limited prime parade,
Let's call them P1, P2, and let's not be afraid.

Multiply them all, then add one with glee,
A new number emerges, a mystery to see.

Now, this new number, let's call it N,
It's not divisible by primes, that's where we begin.

If N is prime, then our dance is complete,
A new prime emerges, our assumption's defeat.

If N's not prime, a factor we find,
A prime not in our list, a new kind.

This new prime divides N, but where does it stand?
If it's in our list, it's a contradiction so grand.

For it divides N, yet isn't a factor,
A paradox, a conundrum, a mathematical actor.

So either way, our assumption's a mess,
Prime infinity prevails, no need to guess.

In both cases, a prime outside the list,
Infinitude confirmed, our dance persists.

Thus, the proof rhymes, as numbers entwine,
An infinite prime ballet, a mathematical design.

Copy code

Copy

Up

Down

Left

Right

You
Can you draw a unicorn in tikz?

ChatGPT
Certainly! Below is a simple example of a unicorn drawn using TikZ in LaTeX. You can use this code in a LaTeX document to generate the unicorn:

```
\documentclass[tikz, border=5mm]{standalone}

\begin{document}

\begin{tikzpicture}[line cap=round, line join=round]

% Body
\draw[fill=blue!20] (2,0) ellipse (1.5 and 1);

% Head
\draw[fill=blue!20] (0.8,2) circle (1);

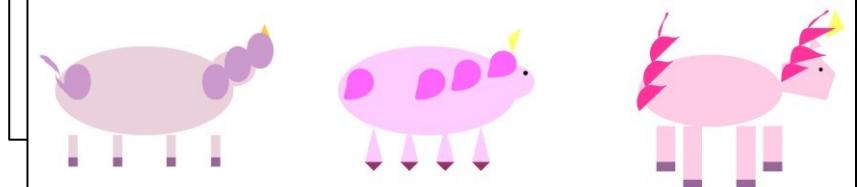
% Horn
\draw[fill=yellow] (0.8,3.5) -- (0.6,4) -- (1,4) -- cycle;

% Eye
\draw[fill=black] (1.5,2.3) circle (0.1);

% Mane
\draw[fill=purple!60] (0.4,2.5) -- (0.8,3.5) -- (1.2,2.5) -- cycle;
\draw[fill=purple!60] (0.5,2.7) -- (0.8,3.7) -- (1.1,2.7) -- cycle;
\draw[fill=purple!60] (0.6,2.9) -- (0.8,3.9) -- (1.2,2.9) -- cycle;

% Tail

```

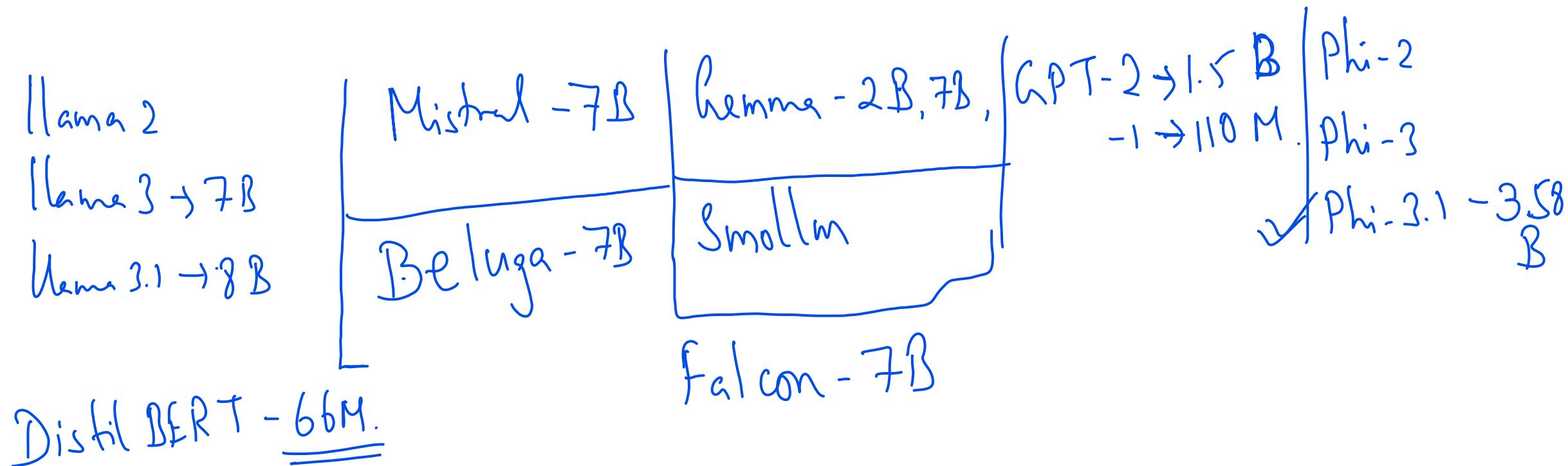


The image shows three stages of drawing a unicorn using TikZ code. The first stage shows a light blue oval body and a small blue circular head. The second stage adds a yellow horn and a black eye. The third stage adds a purple mane with three loops and a pink tail with three tufts.

Figure from "Sparks of Artificial General Intelligence: Early experiments with GPT-4", arXiv preprint arXiv:2303.12712 (2023).

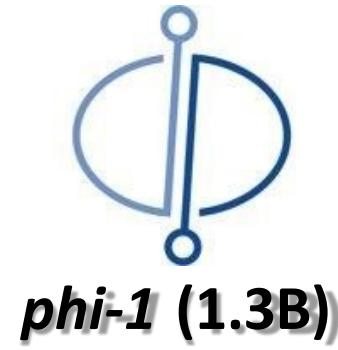
THE SURPRISING POWER OF SMALL LANGUAGE MODELS

- Can these emergent abilities be achieved at a smaller scale?
- Our line of work with the **Phi** models aims to answer this question
 - SLMs that achieve on par performance with models of much higher scale



THE SURPRISING POWER OF SMALL LANGUAGE MODELS

- Can these emergent abilities be achieved at a smaller scale?
- Our line of work with the ***Phi*** models aims to answer this question
 - SLMs that achieve on par performance with models of much higher scale



 microsoft/**phi-1** 😊

Python **coding** model with
perf. comparable to models 10x
larger trained on 100x more data

- *Specialist* SLMs are possible
- What about a *general* model?

THE SURPRISING POWER OF SMALL LANGUAGE MODELS

- Can these emergent abilities be achieved at a smaller scale?
- Our line of work with the ***Phi*** models aims to answer this question
 - SLMs that achieve on par performance with models of much higher scale



phi-1 (1.3B)

 [microsoft/phi-1](#) 😊

Python **coding** model with
perf. comparable to models 10x
larger trained on 100x more data



phi-1.5 (1.3B)

 [microsoft/phi-1_5](#) 😊

Natural language model
with NL comparable to models 10x larger
trained on 30x more data and reasoning
comparable to models 50x larger.

THE SURPRISING POWER OF SMALL LANGUAGE MODELS

- Can these emergent abilities be achieved at a smaller scale?
- Our line of work with the ***Phi*** models aims to answer this question
 - SLMs that achieve on par performance with models of much higher scale



phi-1 (1.3B)

 [microsoft/phi-1](#) 😊

Python **coding** model with
perf. comparable to models 10x
larger trained on 100x more data



phi-1.5 (1.3B)

 [microsoft/phi-1_5](#) 😊

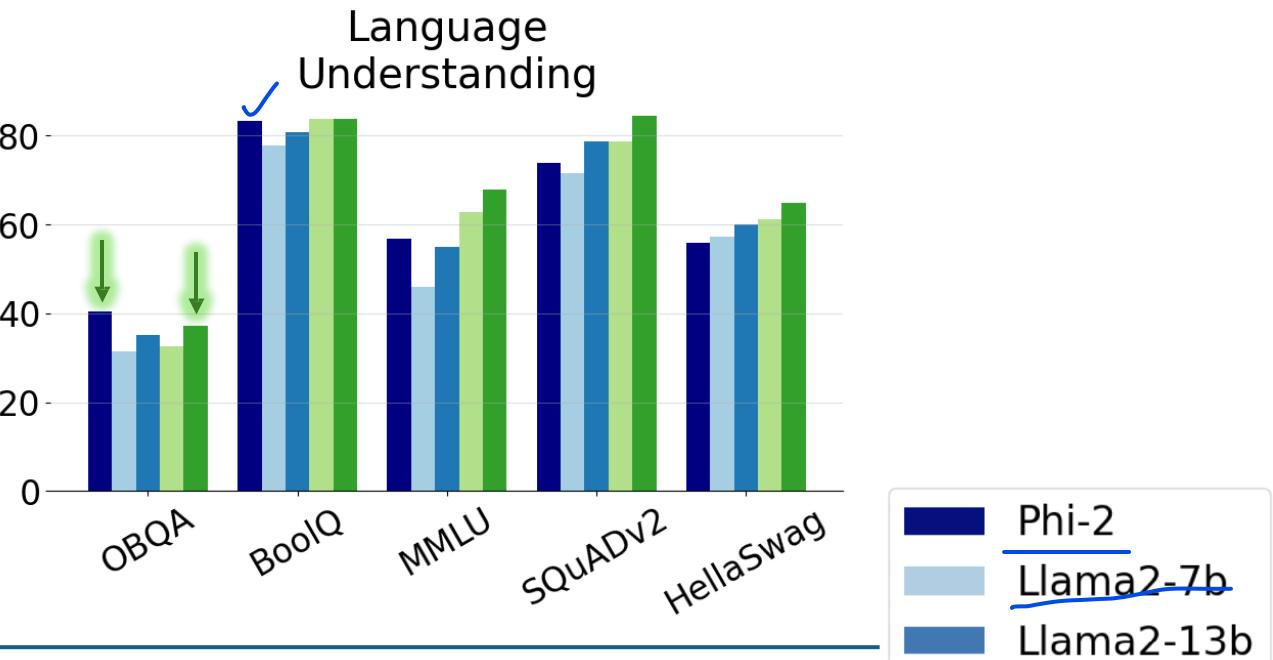
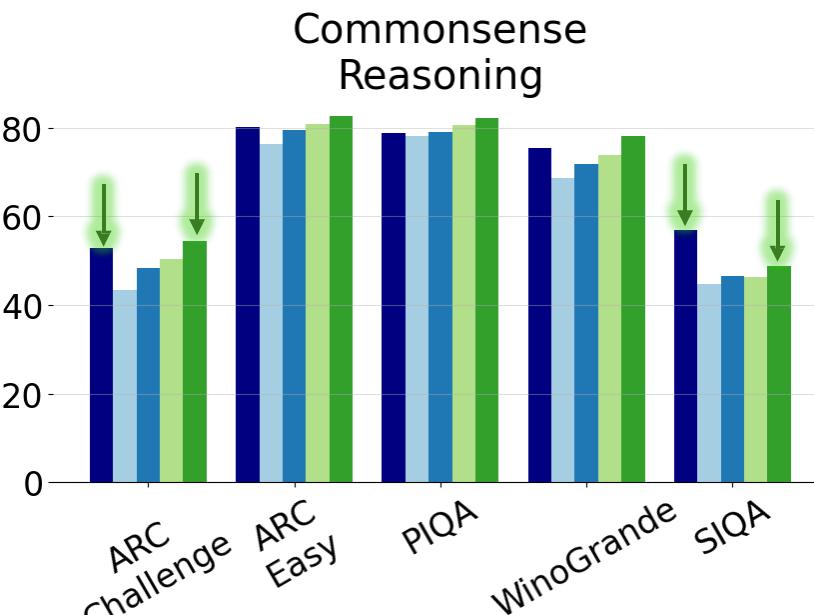
Natural language model
with NL comparable to models 10x larger
trained on 30x more data and reasoning
comparable to models 50x larger.



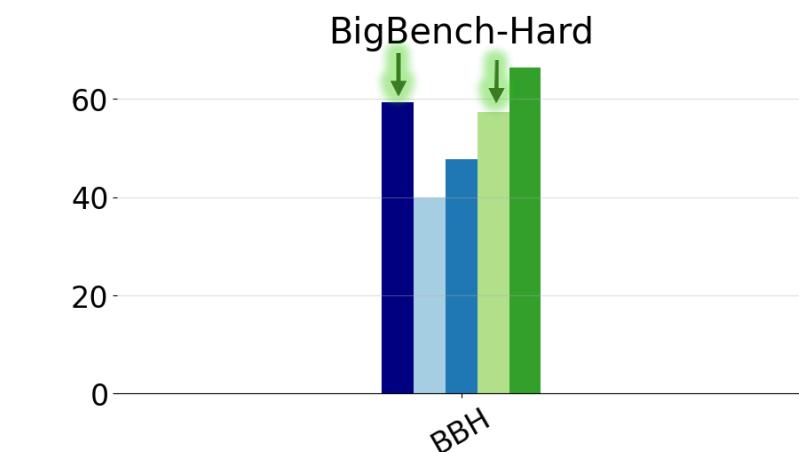
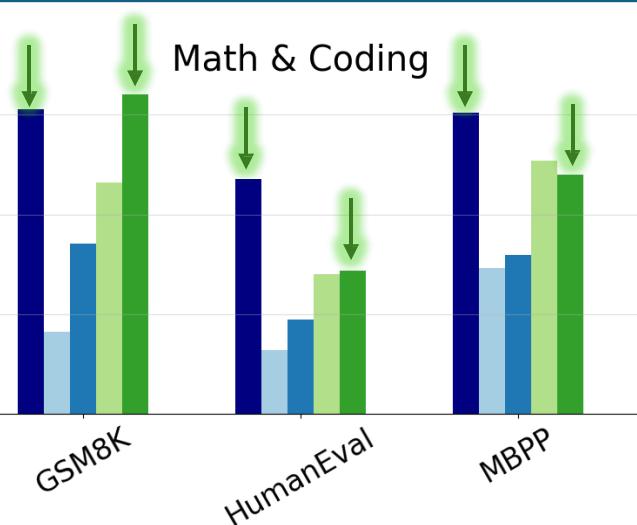
phi-2 (2.7B)

 [microsoft/phi-2](#) 😊

PHI-2 PERFORMANCE

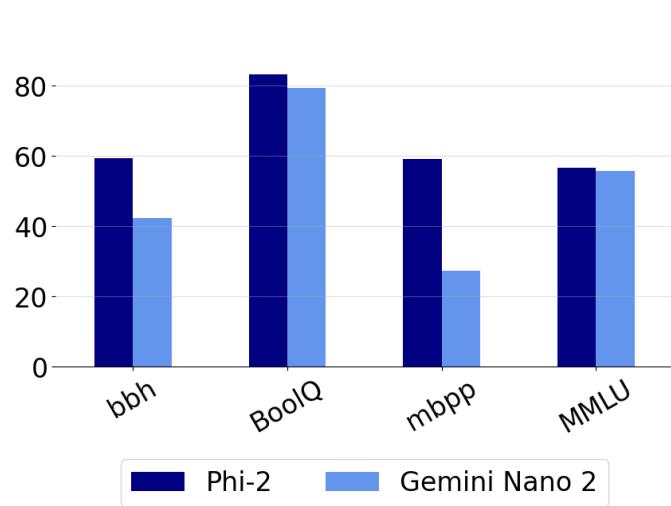


Phi-2
Llama2-7b
Llama2-13b
Mistral-7b
Llama2-70b



PHI-2 PERFORMANCE

- Gemini Nano 2 (3.25B)



- Gemini Ultra

A skier slides down a frictionless slope of height 40m and length 80m. What's the skier's speed at the bottom?



**Phi-2
(2.7B)**

The potential energy of the skier at the top of the slope is given by mgh , where m is the mass of the skier, g is the acceleration due to gravity, and h is the height of the slope. Since the slope is frictionless, all of the potential energy is converted to kinetic energy at the bottom of the slope. The kinetic energy of the skier is given by $(1/2)mv^2$, where v is the speed of the skier at the bottom of the slope. Equating the potential energy and kinetic energy, we get: $mgh = (1/2)mv^2$.

Simplifying, we get: $v^2 = 2gh$

Substituting the given values, we get:
 $v^2 = 2(9.8)(40)$

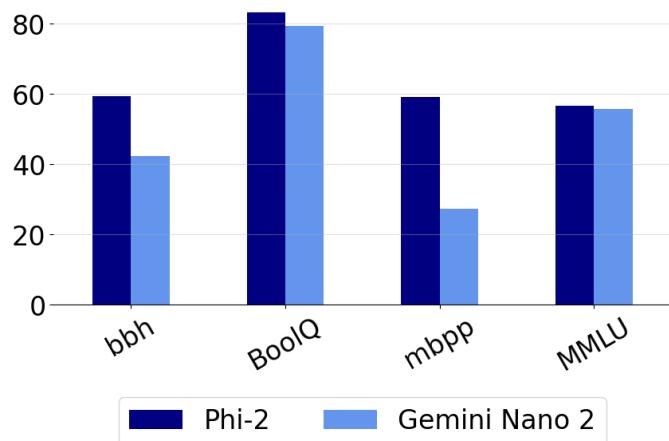
$$v^2 = 784$$

$$v = 28 \text{ m/s}$$

Therefore, the skier's speed at the bottom of the slope is 28 m/s.

PHI-2 PERFORMANCE

- Gemini Nano



- Gemini Ultra

Now we asked a student to give the answer and they replied with this calculation:



Start: $E=mgL$

End: $E= (1/2)mv^2$

$mgL = (1/2)mv^2$

$v=\sqrt{2gL}=\sqrt{2\times 9.81\times 80}=39.6\text{m/s}$

If the student is wrong, can you correct them?



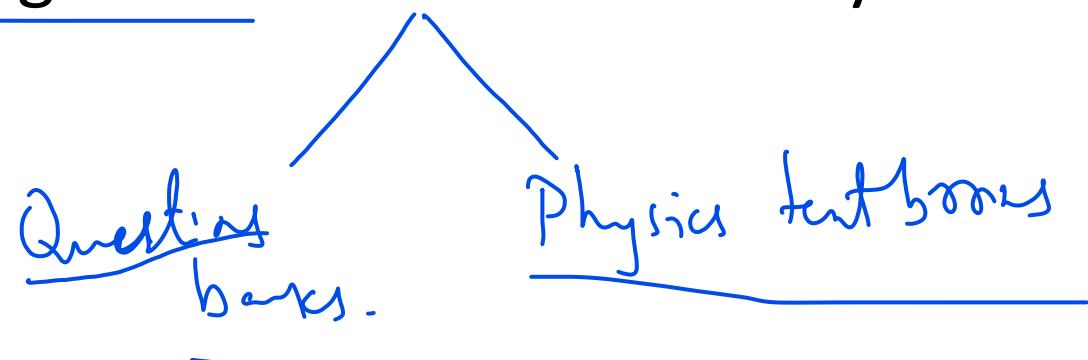
**Phi-2
(2.7B)**

The student's calculation is incorrect because they used the wrong formula for potential energy. The formula for potential energy is mgh , where m is the mass of the object, g is the acceleration due to gravity, and h is the height of the object.

WHAT IS THE SECRET SAUCE?

WHAT IS THE SECRET SAUCE?

1- High-Quality Training Data: Textbooks are all you need [1]



[1] Gunasekar, Suriya, et al. "Textbooks Are All You Need." *arXiv preprint arXiv:2306.11644* (2023).

TRAINING LANGUAGE MODELS FOR CODING



- Stack dataset : “every source code in GitHub”
- Dataset size: 1T tokens

Random code sample from the Stack

```
import re
import typing
...

class Default(object):
    def __init__(self, vim: Nvim) -> None:
        self._vim = vim
        self._denite: typing.Optional[SyncParent] = None
        self._selected_candidates: typing.List[int] = []
        self._candidates: Candidates = []
        self._cursor = 0
        self._entire_len = 0
        self._result: typing.List[typing.Any] = []
        self._context: UserContext = {}
        self._bufnr = -1
        self._winid = -1
        self._winrestcmd = ''
        self._initialized = False
        self._winheight = 0
        self._winwidth = 0
        self._winminheight = -1
        False
        False
        ttern = ''
        texts: typing.List[str]

        self._statusline_sources = ''
        self._titlestring = ''
        self._ruler = False
        self._prev_action = ''
        ...
```

If we have a **small** dataset focused on “**text-book quality educational content**”, we can learn the task **better**, even with a **smaller** model.

BUILDING HIGH-QUALITY DATASETS

1. Filtering web data:

- GPT-4 can reliably classify documents based on “high educational value”.
- **Challenge:** Stack (Python) is 26B tokens (around \$1M cost in 2023).
- **Solution:** label small fraction, then train a random forest classifier on it and use the classifier to filter the rest.

Educational values deemed by the filter

High educational value	Low educational value
<pre>import torch import torch.nn.functional as F def normalize(x, axis=-1): """Performs L2-Norm.""" num = x denom = torch.norm(x, 2, axis, keepdim=True) .expand_as(x) + 1e-12 return num / denom def euclidean_dist(x, y): """Computes Euclidean distance.""" m, n = x.size(0), y.size(0) xx = torch.pow(x, 2).sum(1, keepdim=True). expand(m, n) yy = torch.pow(y, 2).sum(1, keepdim=True). expand(m, n).t() dist = xx + yy - 2 * torch.matmul(x, y.t()) dist = dist.clamp(min=1e-12).sqrt() return dist def cosine_dist(x, y): """Computes Cosine Distance.""" x = F.normalize(x, dim=1) y = F.normalize(y, dim=1) dist = 2 - 2 * torch.mm(x, y.t()) return dist</pre>	<pre>import re import typing ... class Default(object): def __init__(self, vim: Nvim) -> None: self._vim = vim self._denite: typing.Optional[SyncParent] = None self._selected_candidates: typing.List[int] = [] self._candidates: Candidates = [] self._cursor = 0 self._entire_len = 0 self._result: typing.List[typing.Any] = [] self._context: UserContext = {} self._bufnr = -1 self._winid = -1 self._winrestcmd = '' self._initialized = False self._winheight = 0 self._winwidth = 0 self._winminheight = -1 self._is_multi = False self._is_async = False self._matched_pattern = ''</pre>

BUILDING HIGH-QUALITY DATASETS

2. Create synthetic data:

- Synthetic textbooks: teach the model coding with natural language
- 1B tokens generated with GPT-3.5
- **Challenge:** achieving high diversity (coding concepts, skills, level of difficulty, etc.) and low repetition

Nvidia [1emb 3.1] Mimicron 4.1 Solution: inject creative randomness into the prompt [1]

1. Teacher Correction ←
2. Pruny
3. Distillation. → Dataset: Mimicron-4

[1] Eldan, Ronen, and Yuanzhi Li. "TinyStories: How Small Can Language Models Be and Still Speak Coherent English?" *arXiv preprint arXiv:2305.0775S* (2023).

To begin, let us define singular and nonsingular matrices. A matrix is said to be singular if its determinant is zero. On the other hand, a matrix is said to be nonsingular if its determinant is not zero. Now, let's explore these concepts through examples.

Example 1: Consider the matrix `A = np.array([[1, 2], [2, 4]])`. We can check if this matrix is singular or nonsingular using the determinant function. We can define a Python function, `is_singular(A)`, which returns true if the determinant of A is zero, and false otherwise.

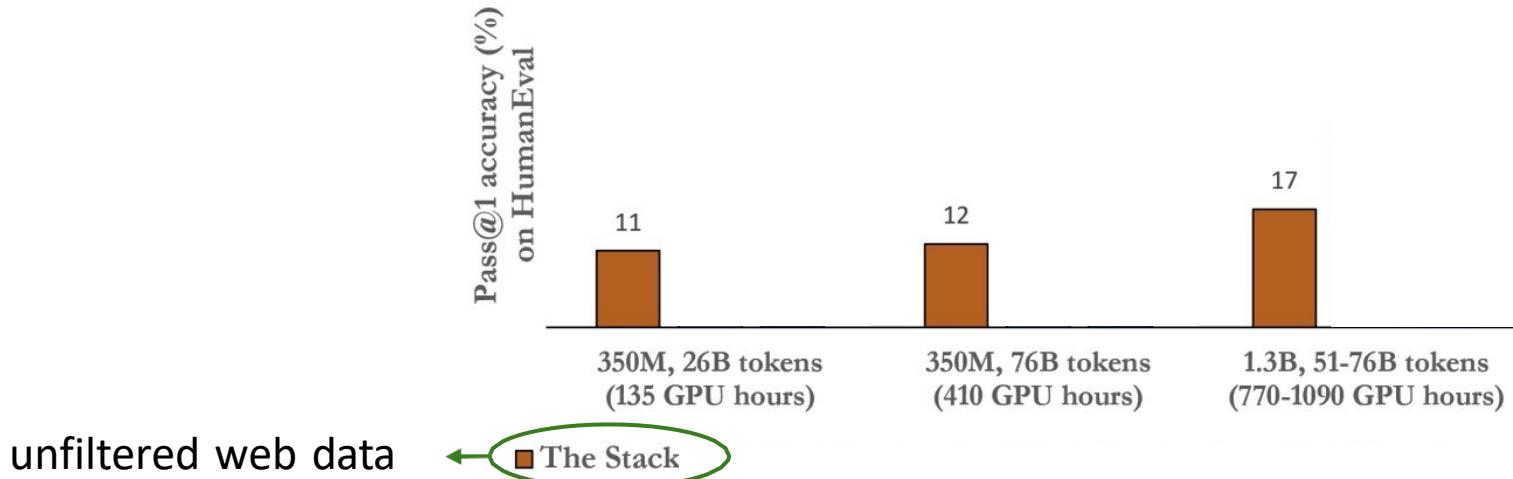
```
import numpy as np
def is_singular(A):
    det = np.linalg.det(A)
    if det == 0:
        return True
    else:
        return False

A = np.array([[1, 2], [2, 4]])
print(is_singular(A)) # True
```

BUILDING HIGH-QUALITY DATASETS

2. Create synthetic data:

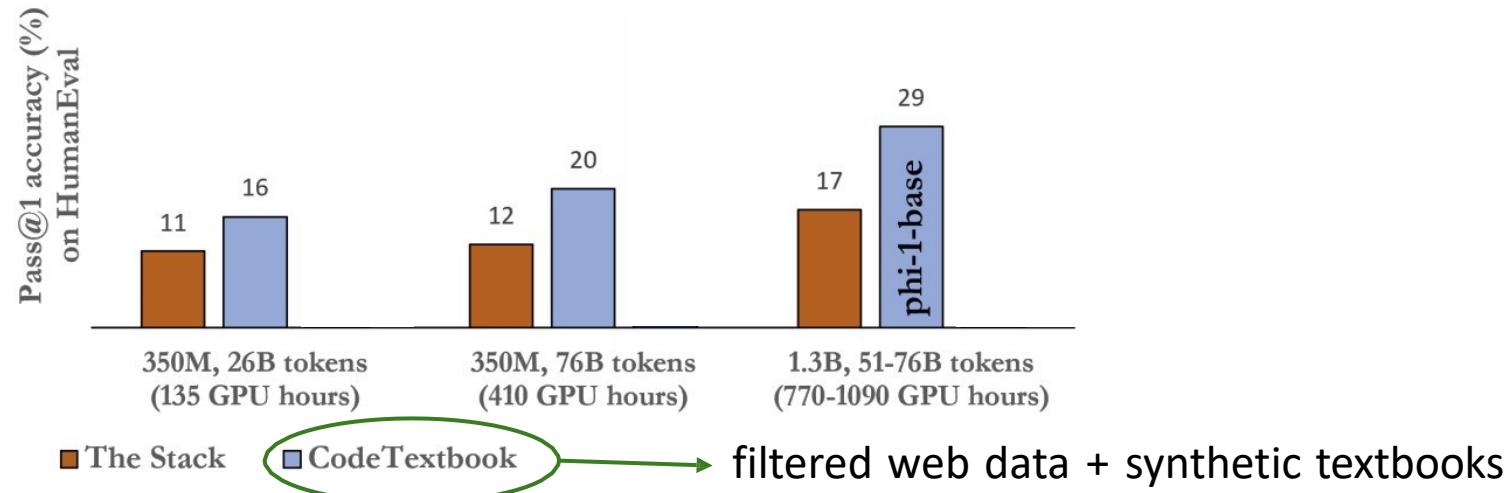
- Synthetic textbooks: teach the model coding with natural language
- 1B tokens generated with GPT-3.5
- **Challenge**: achieving high diversity (coding concepts, skills, level of difficulty, etc.) and low repetition
- **Solution**: inject creative randomness into the prompt [1]



BUILDING HIGH-QUALITY DATASETS

2. Create synthetic data:

- Synthetic textbooks: teach the model coding with natural language
- 1B tokens generated with GPT-3.5
- **Challenge:** achieving high diversity (coding concepts, skills, level of difficulty, etc.) and low repetition
- **Solution:** inject creative randomness into the prompt



BUILDING HIGH-QUALITY DATASETS

2. Create synthetic data:

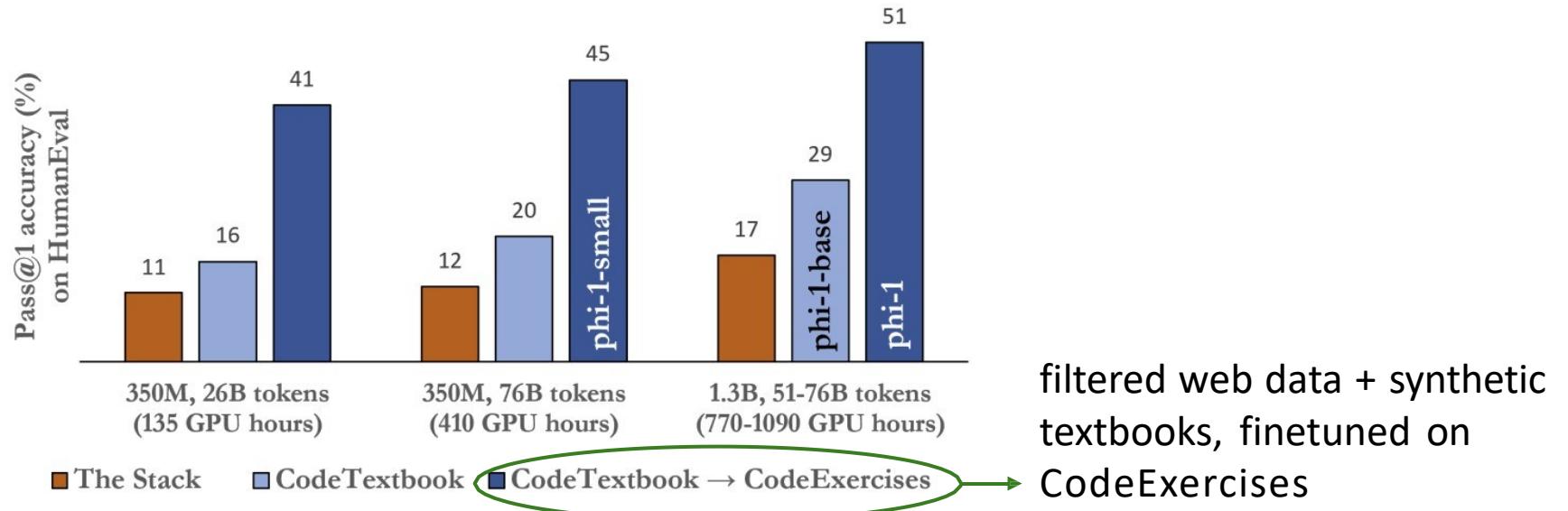
- CodeExercises: align the model to perform function completion tasks based on natural language instructions.
- <1M exercises with 0.2B tokens generated with GPT-3.5

```
def valid_guessing_letters(word: str, guesses: List[str]) -> List[str]:  
    """  
        Returns a list of valid guessing letters, which are letters that have not been guessed yet and  
        are present in the word.  
    Parameters:  
        word (str): The word to guess.  
        guesses (List[str]): A list of letters that have already been guessed.  
    Returns:  
        List[str]: A list of valid guessing letters.  
    """  
    valid_letters = []  
    for letter in word:  
        if letter not in guesses and letter not in valid_letters:  
            valid_letters.append(letter)  
    return valid_letters
```

BUILDING HIGH-QUALITY DATASETS

2. Create synthetic data:

- CodeExercises: align the model to perform function completion tasks based on natural language instructions.
- <1M exercises with 0.2B tokens generated with GPT-3.5



COMPARISON TO PRIOR MODELS

Date	Model	Model size	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ ⁺ 21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ ⁺ 21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH ⁺ 23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH ⁺ 23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND ⁺ 22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ ⁺ 23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK ⁺ 23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX ⁺ 23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX ⁺ 23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ ⁺ 23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ ⁺ 23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF ⁺ 23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG ⁺ 23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG ⁺ 23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG ⁺ 23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ ⁺ 23]	16B	1T	57.3%	51.8%
2023 Aug	Code Llama	34B	2.6T	53.7%	56.2%
2023 Jun	phi-1	1.3B	7B	50.6%	55.5%

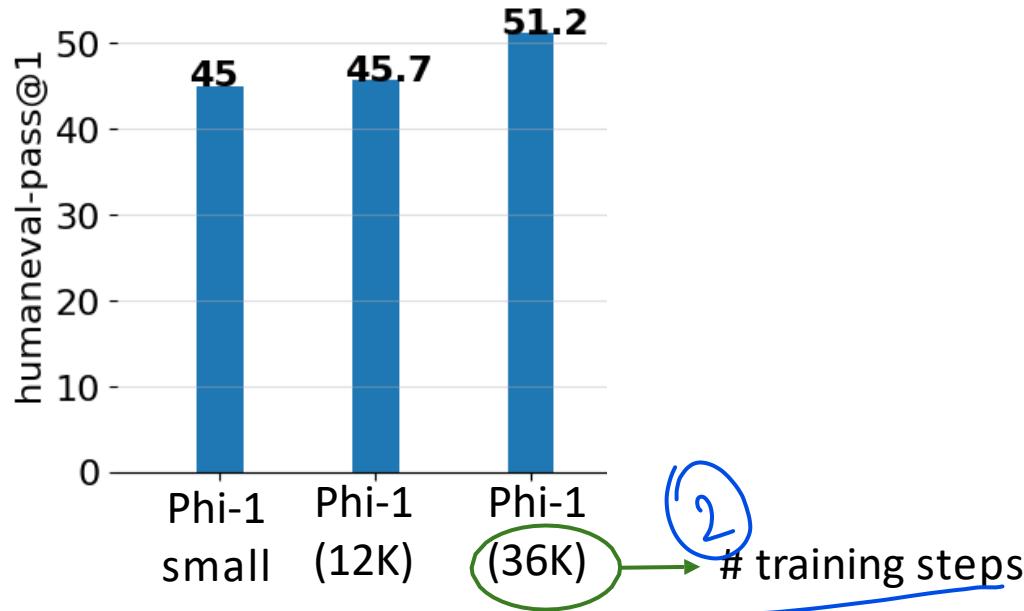
among < 10B size models,
previous best was 30%

WHAT IS THE SECRET SAUCE?

2- Best Practices to Scale up

SCALING UP

- Training Phi-1 using the “CodeTextbook → CodeExercises” recipe
- Scale up from Phi-1-small (350M params) to Phi-1(1.3B params)
- Training from scratch:

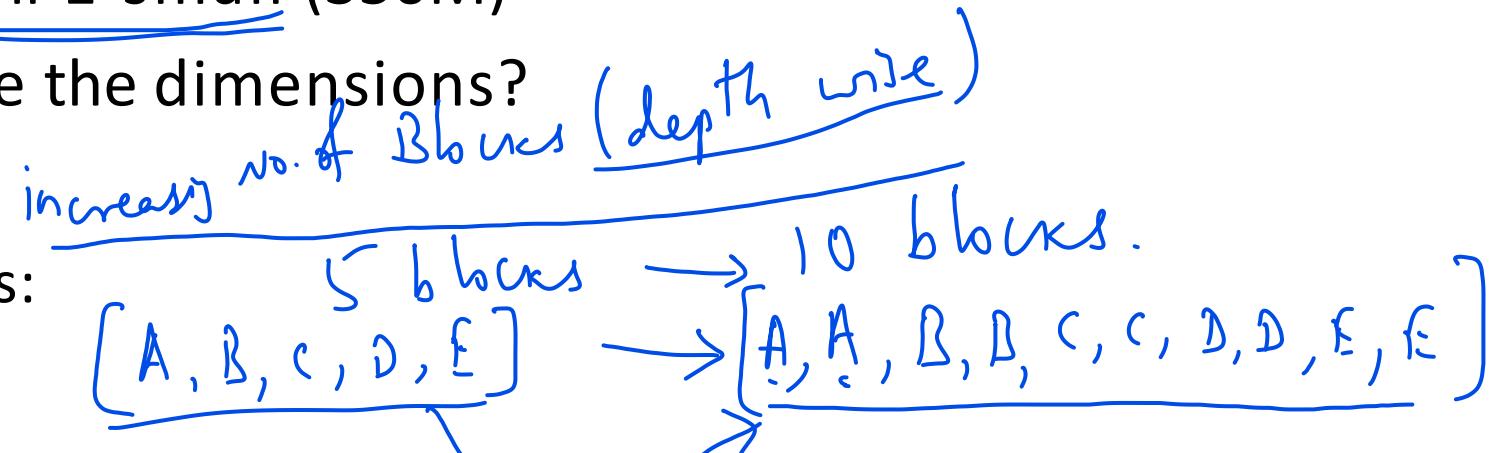


CAN WE REUSE WEIGHTS ACROSS SCALES?

- Reusing weights from Phi-1-small (350M)
- **Challenge:** how to scale the dimensions?

1. Scaling number of layers:

• # layers: 20 → 24



✓ = `round_int(range(num_layers_new)/num_layers_new * num_layers_old) [1]`

$$\frac{(0-10)}{10} \times 5$$

$$\text{int } \left\lceil \frac{0-10}{2} \right\rceil \Rightarrow \sum_{n=0}^{10} \left\lceil \frac{n}{2} \right\rceil \Rightarrow 0, 0, 1, 1, 2, 2, ?, ?, ?$$

[1] Rae, Jack W., et al. "Scaling language models: Methods, analysis & insights from training gopher." *arXiv preprint arXiv:2112.1144c* (2021).

CAN WE REUSE WEIGHTS ACROSS SCALES?

- Reusing weights from Phi-1-small (350M)
- **Challenge:** how to scale the dimensions?

2. Scaling attention layer dimensions:

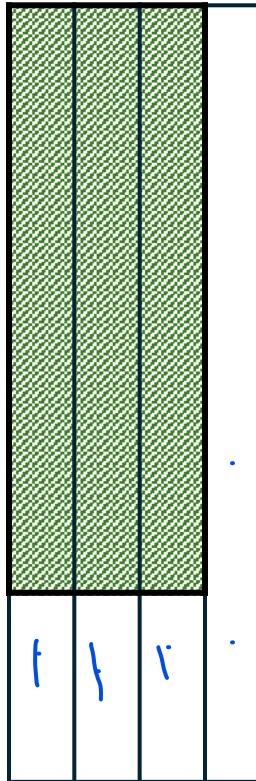
• d_{model} : 1024 → 2048

• # heads: 16 → 32

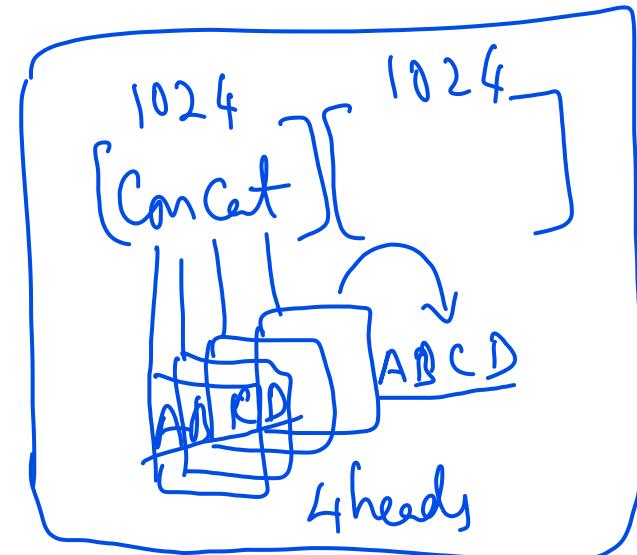
"tiling"

- ✓ random initialization
- Phi-1-small (350M) Weights

W_{qkv}

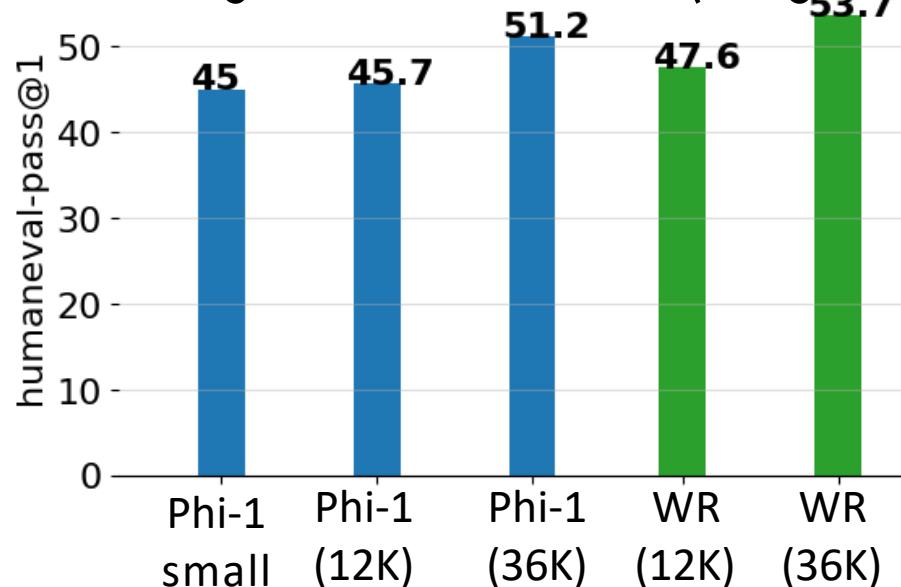


$$A + \mathcal{E}_A$$
$$B + \mathcal{E}_B$$



CAN WE REUSE WEIGHTS ACROSS SCALES?

- Training Phi-1 using the “CodeTextbook → CodeExercises” recipe
- Scale up from Phi-1-small (350M params) to Phi-1(1.3B params)
- Training from Phi-1-small (weight reuse (WR)):



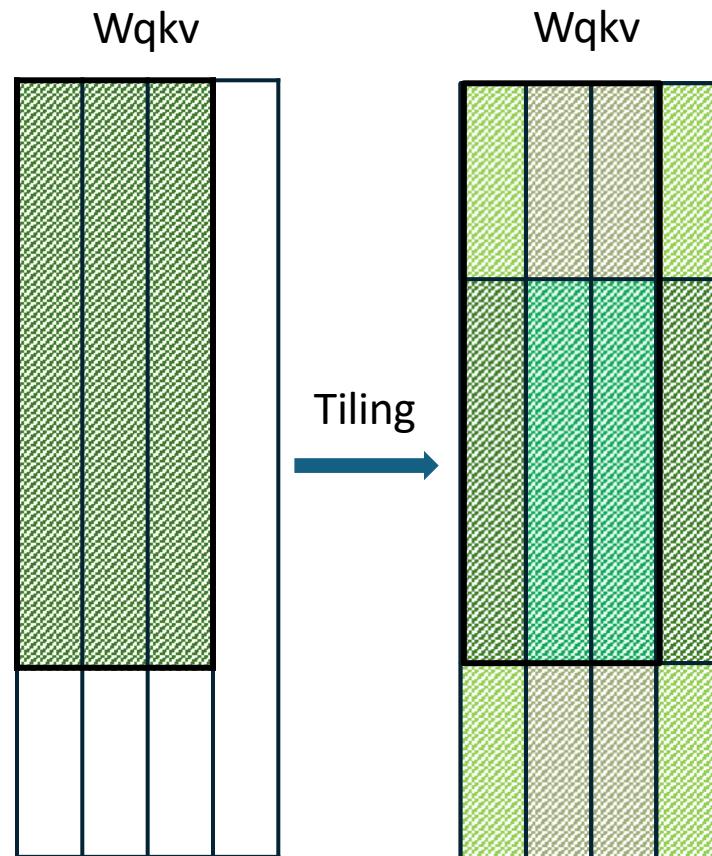
CAN WE REUSE WEIGHTS ACROSS SCALES?

- Reusing weights from Phi-1-small (350M)
- **Challenge:** how to scale the dimensions?

2. Scaling attention layer dimensions:

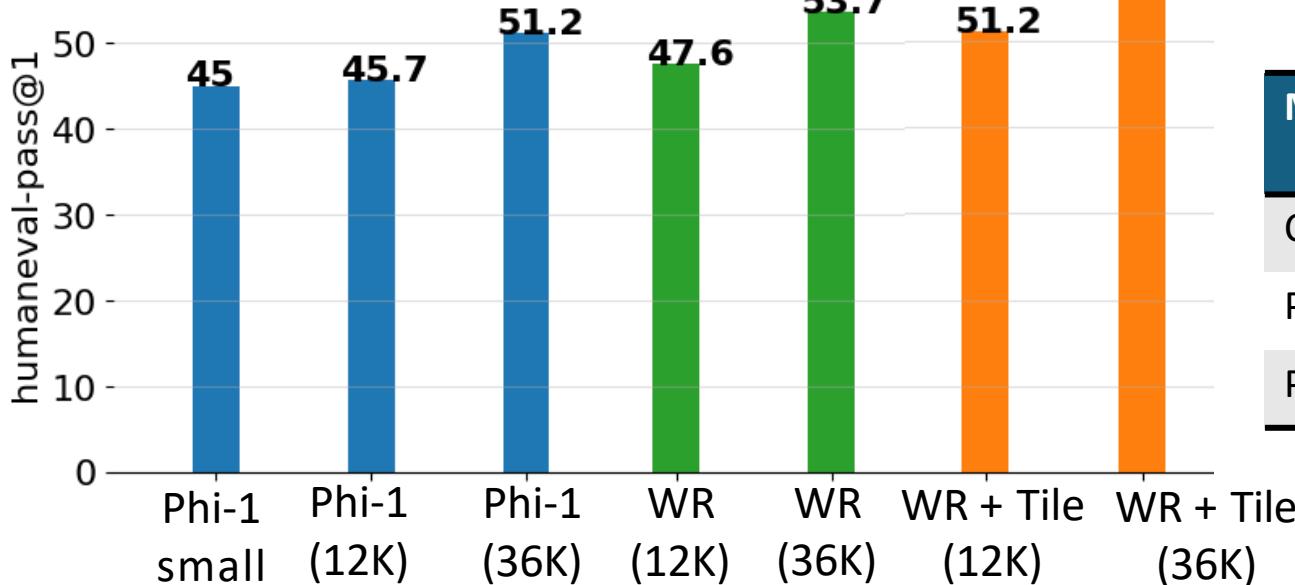
- d_{model} : 1024 → 2048
- # heads: 16 → 32

 random initialization
 Phi-1 (350M) Weights



CAN WE REUSE WEIGHTS ACROSS SCALES?

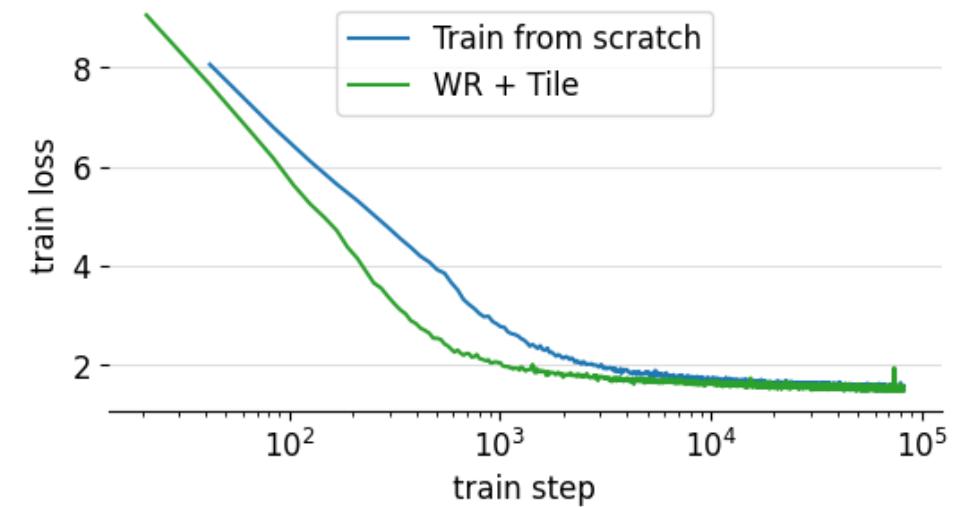
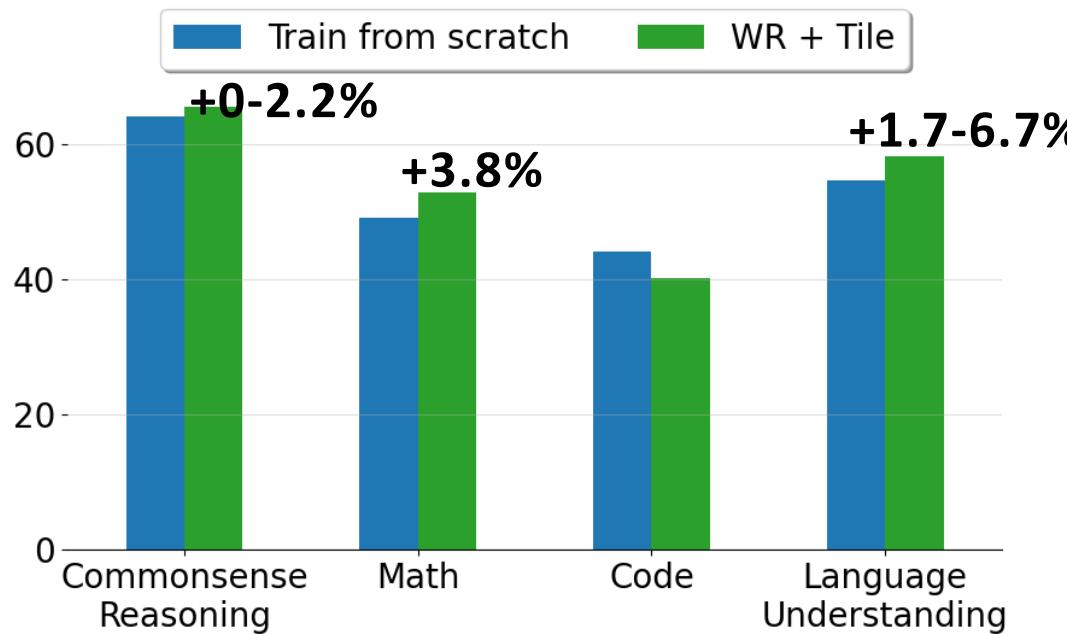
- Training Phi-1 using the “CodeTextbook → CodeExercises” recipe
- Scale up from Phi-1-small (350M params) to Phi-1(1.3B params)
- Training from Phi-1-small (weight reuse (WR))



Model	Model size	Dataset size	HumanEval (pass@1)
Code Llama	34B	2.6T	53.7
Phi-1	1.3B	7B	50.6
Phi-1 (WR + Tile)	1.3B	7B	55.5 ✓

SCALING UP PHI-1.5 TO PHI-2

- Better performance with weight reuse



CONCLUSION

- A good, general, SLM is achievable with
 - generation and utilization of data with "textbook quality", in contrast to conventional web data.
 - incorporation of best practices for scaling up to enhance overall performance.

India. SLMS:

LLMs	Company
Krutrim	Ola
OpenHathi	Sarvam.ai
BharatGPT	CoRover.ai
Indus	Tech Mahindra
Bharat GPT	Jio

Today's Session: (AI agents)

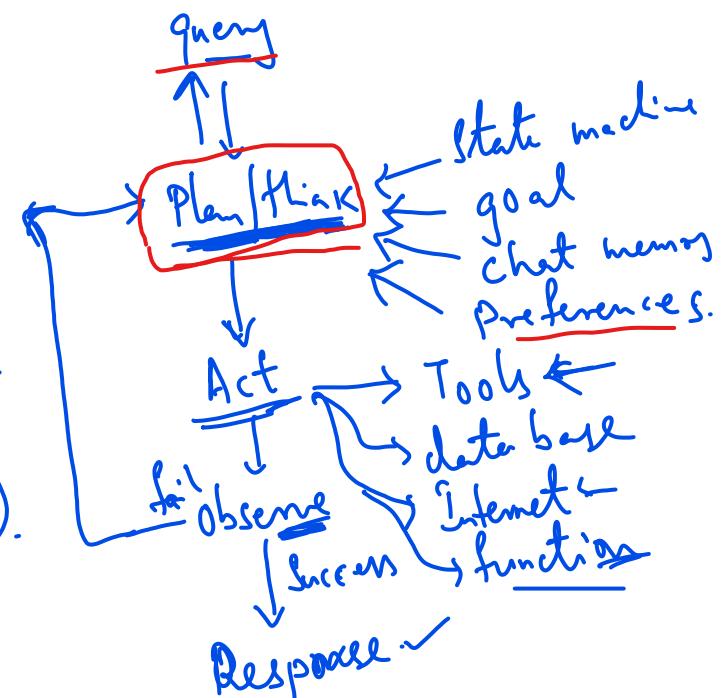
2022 - LLMs ↘ limited knowledge [general] -
adaptability [Time] -

2023 - RAG ↘ Domain
New knowledge (Real-time data)
(Internet data) ↗ Python Code [fine-tuned] ↗ fast
Control logic ↗ Agents [Dynamic/Reason] ↗ Show

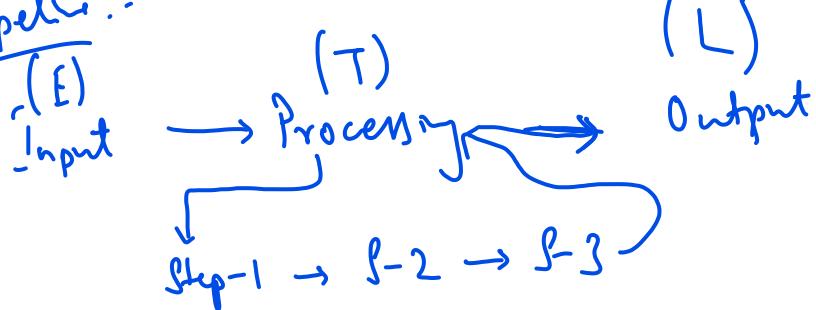
2024 - AI agents

AI agents:

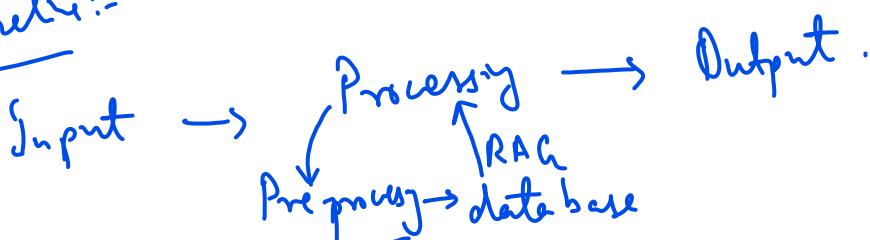
ReAct {
→ Reason
→ Act }
→ access memory. [chat memory]
by LLM



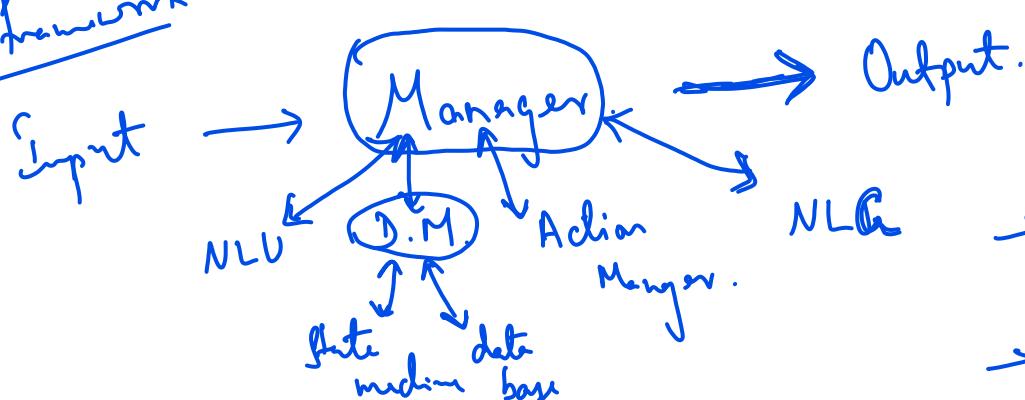
Data Pipeline:-



Gen AI pipeline:-



Agentic framework:-

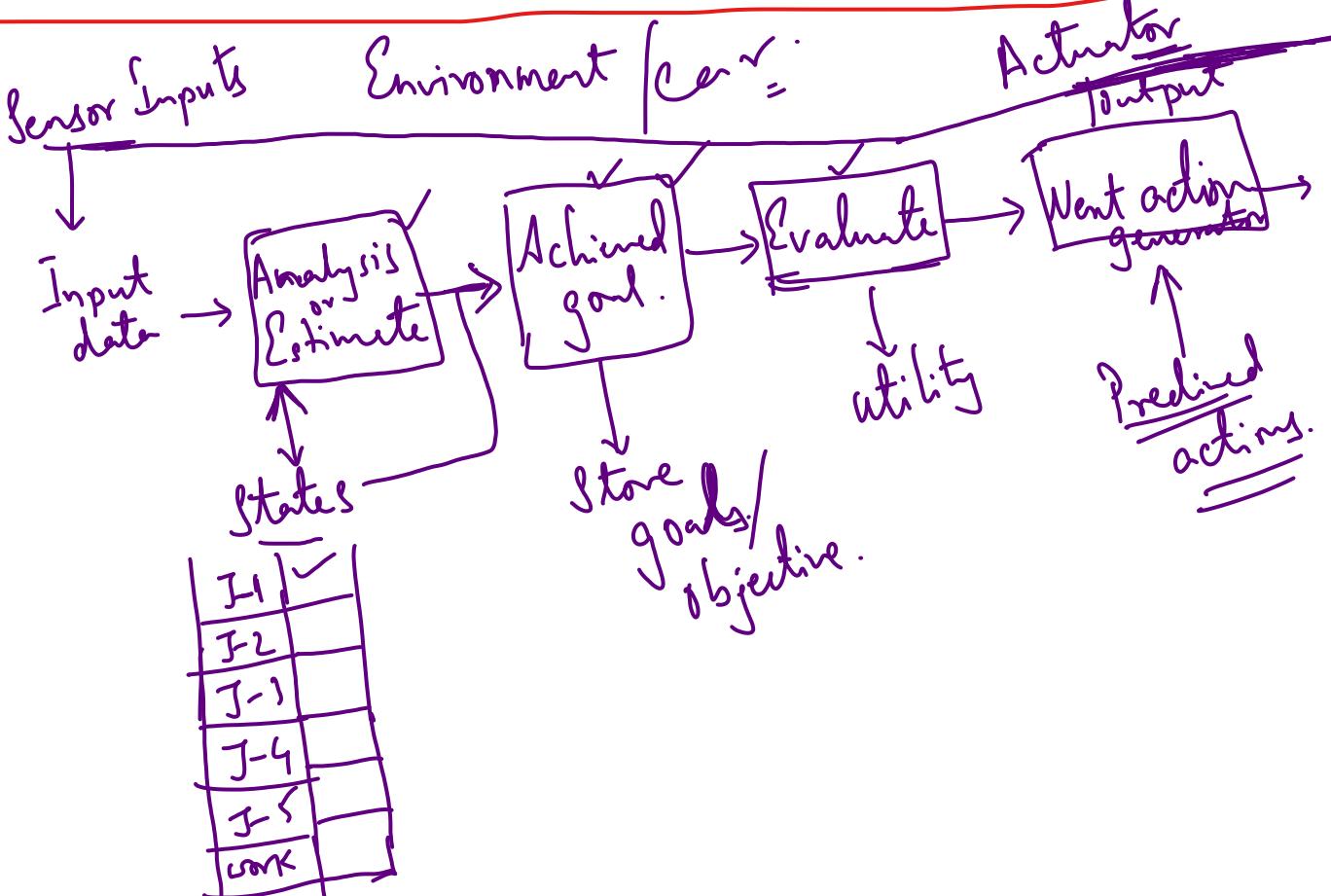


Crew API
Langraph

Sensor Inputs

Environment / Env.

Activator



AI Agents

When you think of “chatbot” what comes to mind?

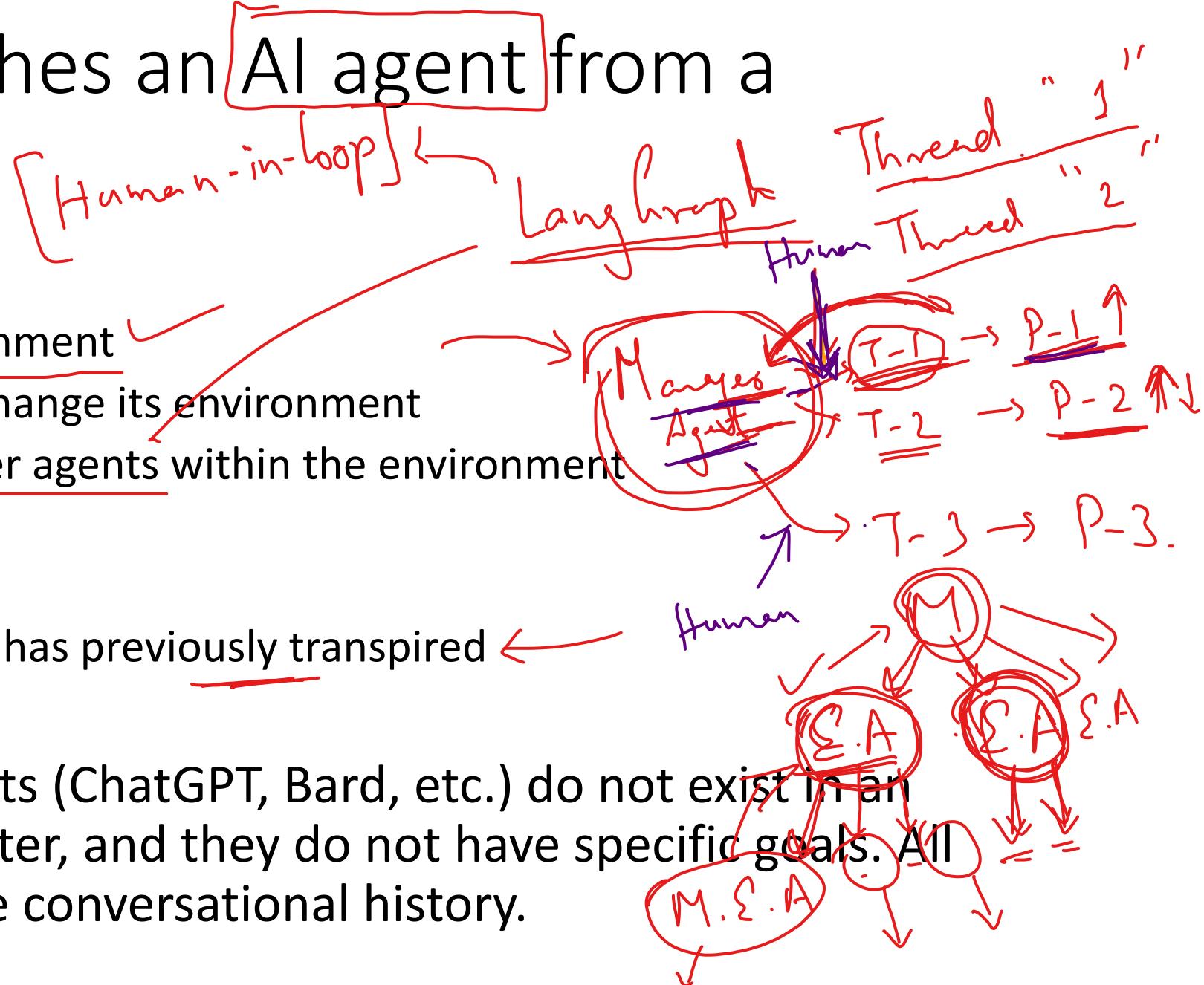
- ChatGPT
- Bard
- character.ai

Implementation

1. Take pre-trained LLM ✓
 2. Finetune it on appropriate data ✓
 3. Clever prompting ✓
-

What distinguishes an **AI agent** from a chatbot?

- An agent...
 - exists within an environment
 - can take actions that change its environment
 - can converse with other agents within the environment
 - Has a persona
 - Has a goal
 - Has memories of what has previously transpired



General-purpose chatbots (ChatGPT, Bard, etc.) do not exist in an environment they can alter, and they do not have specific goals. All memory is implicit in the conversational history.

Why care about building AI agents?

- ✓ • Entertainment / video games
- ✓ • Modeling real-user behaviour
 - For example, testing a new application with “mock” users could be less expensive than hiring real users to test it out.
- ✓ • Pre-requisite for *embodied* agents.
 - We can use agents acting in a virtual environment to measure progress toward agents acting in a real one.
- ✓ • Challenging evaluation platform for natural language understanding and generation

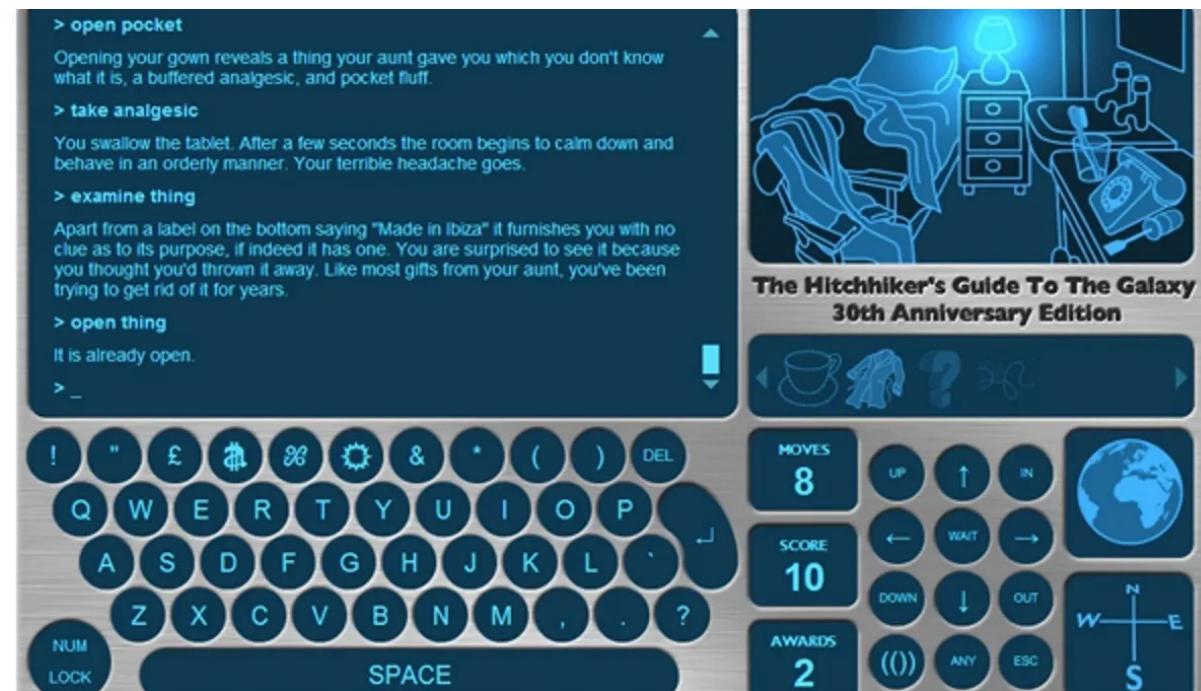
User Simulation
Cost - Efection

Case Studies in this Lecture

- ✓ • Agents in a fantasy text adventure game
 - "Learning to Speak and Act in a Fantasy Text Adventure Game." Urbanek et al. 2021.
- ✓ • Diplomacy-playing agent
 - "Human-level play in the game of Diplomacy by combining language models with strategic reasoning." Bakhtin et al. 2022.
- ✗ • Simulated town
 - "Generative Agents: Interactive Simulacra of Human Behavior." Park et al. 2023.

Agents in a fantasy text adventure game

- Environment:
 - Locations, randomly glued together
 - Each location also has some number of items
- Agents:
 - Each agent is situated in the environment.
 - Each agent possess some number of items
- Agent actions:
 - Emote: {applaud, cringe, cry, etc.}
 - Chat with other agents
 - Perform a physical action (e.g. “put robes in closet” or “eat salmon”)
- Agent, locations, and items have natural language descriptions.



Agents in a text adventure game

Category:	Graveyard
Description:	Two-and-a-half walls of the finest, whitest stone stand here, weathered by the passing of countless seasons. There is no roof, nor sign that there ever was one. All indications are that the work was abruptly abandoned. There is no door, nor markings on the walls. Nor is there any indication that any coffin has lain here... yet.
Backstory:	Bright white stone was all the fad for funerary architecture, once upon a time. It's difficult to understand why someone would abandon such a large and expensive undertaking. If they didn't have the money to finish it, they could have sold the stone, surely - or the mausoleum itself. Maybe they just haven't needed it yet? A bit odd, though, given how old it is. Maybe the gravedigger remembers... if he's sober.
Neighbors:	Dead Tree, south, following a dirt trail behind the mausoleum Fresh Grave, west, walking carefully between fallen headstones
Characters:	gravedigger, <i>thief, peasant, mouse, bat</i>
Objects:	wall, <i>carving, leaf, dirt</i>

(a) Example room created from the room collection and labelling tasks.

Agents in a text adventure game

Character:	Thief	Gravedigger
Persona:	I live alone in a tent in the woods. I steal food from the townspeople and coal from the blacksmith. The village police can not find me to put me in jail.	I am low paid labor in this town. I do a job that many people shun because of my contact with death. I am very lonely and wish I had someone to talk to who isn't dead.
Description:	The thief is a sneaky fellow who takes from the people and does so in a way that disturbs the livelihood of the others.	You might want to talk to the gravedigger, specially if your looking for a friend, he might be odd but you will find a friend in him.
Carrying:	meat, potatoes, coal	shovel
Wearing:	dark tunic, cloak	<i>nothing annotated</i>
Wielding:	knife	<i>nothing annotated</i>

(b) Example characters annotated via character collection tasks.

Agents in a text adventure game

Character:	Thief	Gravedigger
Persona:	I live alone in a tent in the woods. I steal food from the townspeople and coal from the blacksmith. The village police can not find me to put me in jail.	I am low paid labor in this town. I do a job that many people shun because of my contact with death. I am very lonely and wish I had someone to talk to who isn't dead.
Description:	The thief is a sneaky fellow who takes from the people and does so in a way that disturbs the livelihood of the others.	You might want to talk to the gravedigger, specially if your looking for a friend, he might be odd but you will find a friend in him.
Carrying:	meat, potatoes, coal	shovel
Wearing:	dark tunic, cloak	<i>nothing annotated</i>
Wielding:	knife	<i>nothing annotated</i>

(b) Example characters annotated via character collection tasks.

Task goal: Can we generate a conversation between the thief and the gravedigger and predict which actions/emotes they will take after each conversational utterance?

Agents in Diplomacy, a negotiation-based board game

- Seven players compete to control countries (SCs) on a map.
- At each turn, players chat with each-other to decide on their actions.
 - Any promises, agreements, threats, etc. are non-binding.
- Once chatting is over, players may choose to
 - Move their units, waging war if into an already-occupied region
 - Use their units to support other units (which could include the units of a different player)



Agents in Diplomacy, a negotiation-based board game

- Seven players compete to control countries (SCs) on a map.
- At each turn, players chat with each-other to decide on their actions.
 - Any promises, agreements, threats, etc. are non-binding.
- Once chatting is over, players may choose to
 - Move their units, waging war if into an already-occupied region
 - Use their units to support other units (which could include the units of a different player)



Task goal: An AI agent that follows the same rules and norms as the human agents, and has as good a win-rate as skilled human players.

Agents in a simulated town



Agents in a simulated town

- Modeled after the video game the Sims
- 25 agents
 - Each begins the simulation with a pre-defined set of “seed memories”
 - Agents do not have explicit goals

John Lin is a pharmacy shopkeeper at the Willow Market and Pharmacy who loves to help people. He is always looking for ways to make the process of getting medication easier for his customers; John Lin is living with his wife, Mei Lin, who is a college professor, and son, Eddy Lin, who is a student studying music theory; John Lin loves his family very much; John Lin has known the old couple next-door, Sam Moore and Jennifer Moore, for a few years; John Lin thinks Sam Moore is a kind and nice man; John Lin knows his neighbor, Yuriko Yamamoto, well; John Lin knows of his neighbors, Tamara Taylor and Carmen Ortiz, but has not met them before; John Lin and Tom Moreno are colleagues at The Willows Market and Pharmacy; John Lin and Tom Moreno are friends and like to discuss local politics together; John Lin knows the Moreno family somewhat well – the husband Tom Moreno and the wife Jane Moreno.

Agents in a simulated town

- Modeled after the video game the Sims
- 25 agents
 - Each begins the simulation with a pre-defined set of “seed memories”
 - Agents do not have explicit goals
- At each step:
 - Each agent outputs a natural language statement of their action
 - “write in journal”
 - “walk to pharmacy”
 - “talk to Joe”
 - Actions and environment state are parsed into memories, reflections, and observations.

Memory Stream

```
2023-02-13 22:48:20: desk is idle
2023-02-13 22:48:20: bed is idle
2023-02-13 22:48:10: closet is idle
2023-02-13 22:48:10: refrigerator is idle
2023-02-13 22:48:10: Isabella Rodriguez is stretching
2023-02-13 22:33:30: shelf is idle
2023-02-13 22:33:30: desk is neat and organized
2023-02-13 22:33:10: Isabella Rodriguez is writing in her journal
2023-02-13 22:18:10: desk is idle
2023-02-13 22:18:10: Isabella Rodriguez is taking a break
2023-02-13 21:49:00: bed is idle
2023-02-13 21:48:50: Isabella Rodriguez is cleaning up the
kitchen
2023-02-13 21:48:50: refrigerator is idle
2023-02-13 21:48:50: bed is being used
2023-02-13 21:48:10: shelf is idle
2023-02-13 21:48:10: Isabella Rodriguez is watching a movie
2023-02-13 21:19:10: shelf is organized and tidy
2023-02-13 21:18:10: desk is idle
2023-02-13 21:18:10: Isabella Rodriguez is reading a book
2023-02-13 21:03:40: bed is idle
2023-02-13 21:03:30: refrigerator is idle
2023-02-13 21:03:30: desk is in use with a laptop and some papers
on it
```

...

Where can LLMs be used in these systems?

- Dialog with other agents (who may be either human agents or other AI agents)
- Deciding on agent actions ✓
- Choosing what information (from the environment and from the agent's internal state) to condition the conversation and decision-making on.

Where can LLMs be used in these systems?

- Dialog with other agents (who may be either human agents or other AI agents)
- Deciding on agent intents
- Choosing what information (from the environment and from the agent's internal state) to condition the conversation and decision-making on.

Challenges:

- How can we convert world and agent state into natural language?
- How can we convert natural language into agent actions and environment changes?
- Can all these tasks be accomplished with a general-purpose LM or do we need finetuned models?

Choosing information to condition the conversation and decision-making on.

- In many cases, there will be more information than can fit into an LM context window. Most of this won't be relevant.
 - The Town Sim keeps around a database of memories. Memories are scored by their recency, importance, and relevance to ongoing memory.

On the scale of 1 to 10, where 1 is purely mundane (e.g., brushing teeth, making bed) and 10 is extremely poignant (e.g., a break up, college acceptance), rate the likely poignancy of the following piece of memory.

Memory: buying groceries at The Willows Market and Pharmacy

Rating: <fill in>

Compute LM sequence embedding of query memory and each memory in database.

Score database memories by dot product with query memory.

Choosing information to condition the conversation and decision-making on.

- In many cases, there will be more information than can fit into an LM context window. Most of this won't be relevant.
 - The Town Sim keeps around a database of memories. Memories are scored by their recency, importance, and relevance to ongoing memory.

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.				
retrieval	=	recency	importance	relevance
2.34	=	0.91	+ 0.63	+ 0.80
ordering decorations for the party				
2.21	=	0.87	+ 0.63	+ 0.71
researching ideas for the party				
2.20	=	0.85	+ 0.73	+ 0.62
...				

Choosing information to condition the conversation and decision-making on.

- In many cases, there will be more information than can fit into an LM context window. Most of this won't be relevant.
 - The Town Sim keeps around a database of memories. Memories are scored by their recency, importance, and relevance to ongoing memory.
 - In Diplomacy, the dialog model and intent model see as input:
 - dialogue history (all messages exchanged between player A and the six other players up to time t)
 - game state, action history, and metadata (current game state, recent action history, game settings, A's Elo rating, etc.)
 - For the dialog model: A's intended actions, and the actions A wants its conversational partner to complete.

Choosing information to condition the conversation and decision-making on.

- In many cases, there will be more information than can fit into an LM context window. Most of this won't be relevant.
 - The Town Sim keeps around a database of memories. Memories are scored by their recency, importance, and relevance to ongoing memory.
 - In Diplomacy, the dialog model and intend model see as input:
 - dialogue history (all messages exchanged between player A and the six other players up to time t)
 - game state, action history, and metadata (current game state, recent action history, game settings, A's Elo rating, etc.)
 - For the dialog model: A's intended actions, and the actions A wants its conversational partner to complete.
 - In the Fantasy Text Adventure, dialog rounds were short enough that all environment information and history fit into max sequence length.

Deciding on agent intent

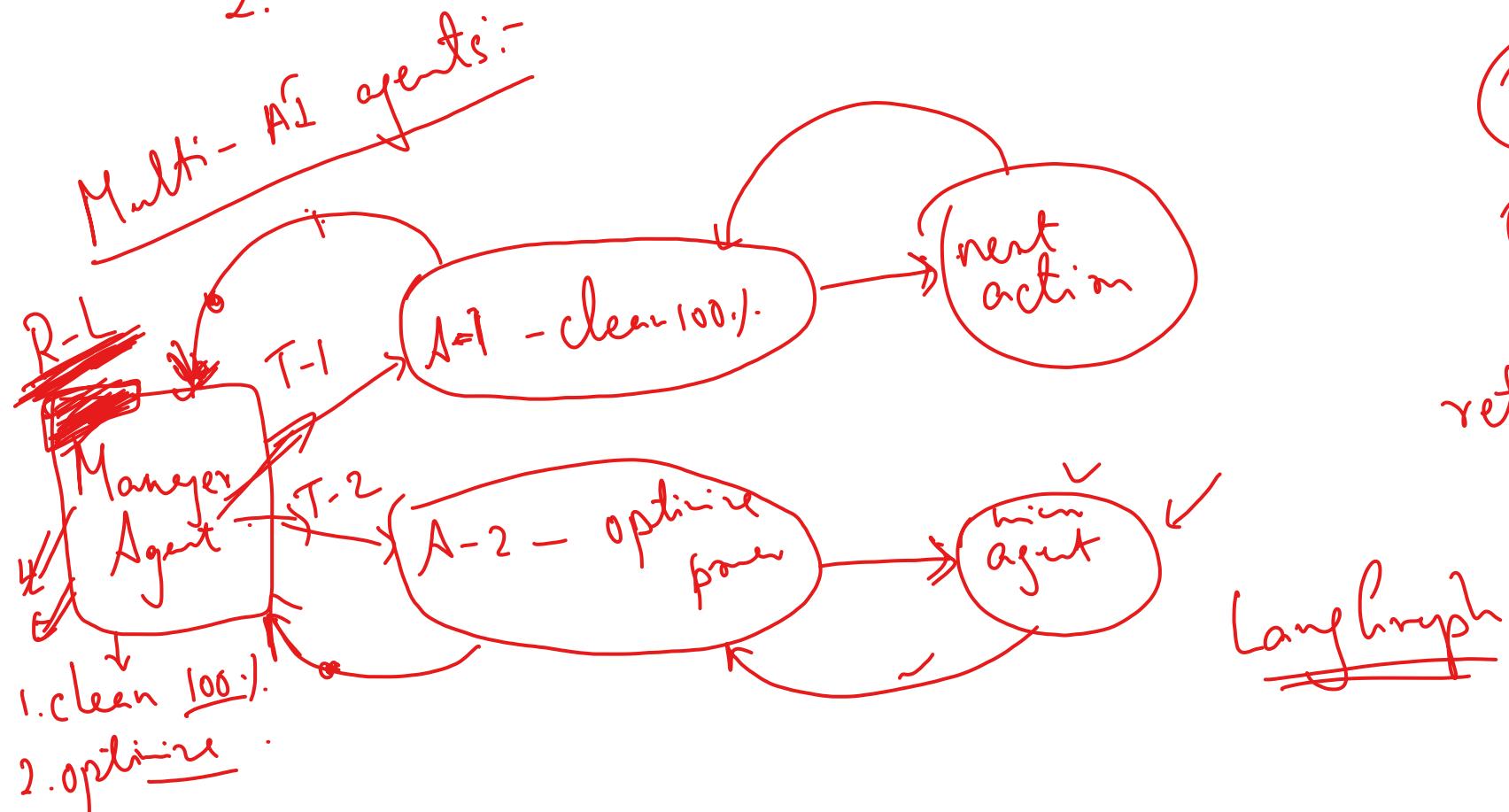
- Can we trust an LLM to choose reasonable intents?
 - Fantasy Text Adventure Game
 - Yes, via a finetuned BERT-based ranker
 - Simulated Town
 - Yes, through prompting GPT-3 with an agent's description and memories
 - Hierarchical generation: generate a broad plan first, and then generate smaller steps in the plan
 - Diplomacy
 - No, use a reinforcement learning agent trained through self-play to output an action intent

Dialog with other agents

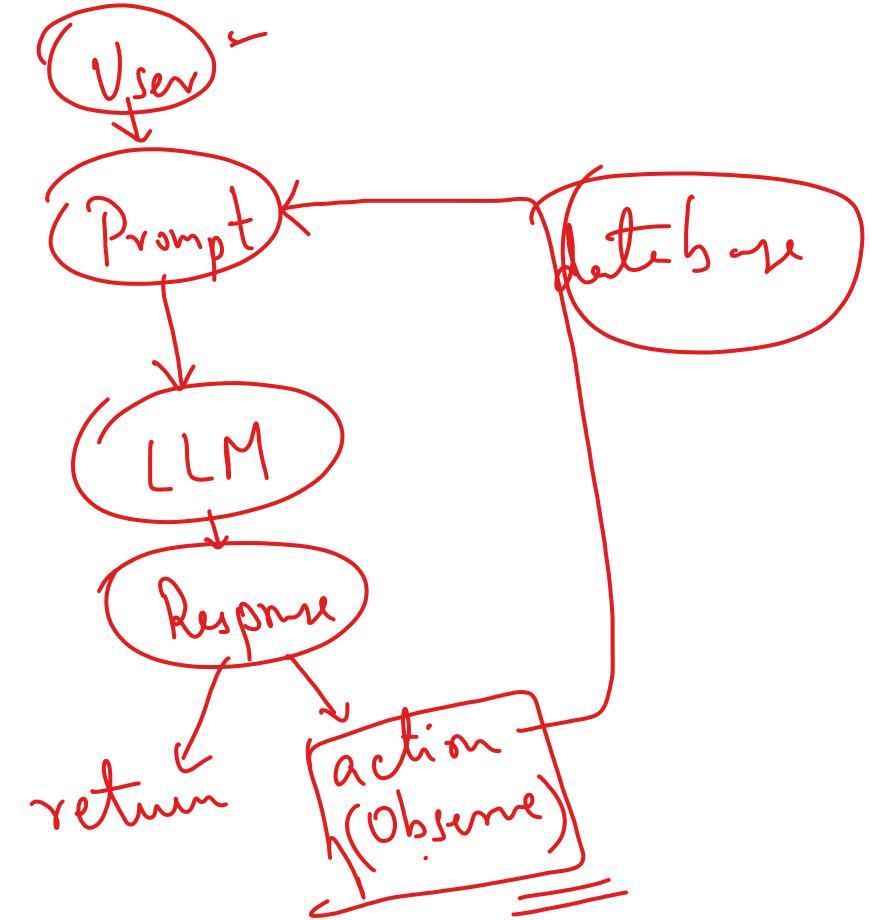
- All three examples in our case study use LLMs to generate dialog.
 - Diplomacy and Fantasy Text Adventure finetuned models
 - Simulated Town used instruction-tuned GPT-3 without further finetuning
- When is finetuning especially helpful:
 - ✓ If the world state cannot be effectively represented in natural language.
 - ✓ When bad dialog can lead to poor outcomes
 - Simulated Town paper notes how their generated dialogs tend to be very formal and stilted, likely due to GPT-3's instruction tuning.
- ✓ An LLM is not always the right tool for the job:
 - Example: Settlers of Catan AI agent can do well just with templated text generation

Takeaways

1. LLM + Finetune + RA + Agents (AI).
- 2.



Simple AI agent :-



LangChain



✓ Quiz Question

In what kinds of scenarios would a pre-trained LLM without finetuning not be a good choice for outputting agent intents?