# Large Language Models and Recommender Systems

Silin DU

*Department of Management Science and Engineering*
*School of Economics and Management*
*Tsinghua University*
`dsl21@mails.tsinghua.edu.cn`

September 22, 2023

## Overview I

In this slide, we first introduce *large language models* (LLMs) following [51], including

- Key concepts and recent developments.
- Pre-training.
- Instruction tuning.
- Alignment tuning.
- Parameter-efficient and memory-efficient model adaptation.
- In-context learning and chain-of-thought prompting.
- Planning for complex task solving.

Then, we focus on LLMs for recommender systems [8].

- LLMs for generative recommendations [23]
- TALLRec
- ChatGPT for recommendations

## Overview II

Finally, we share LLMs for job recommendations.

- RecruitmentPro
- Generative Job Recommendations: GIRL

## Contents

## Language Model

*Language modeling* (LM) is one of the major approaches to advancing language intelligence of machines.

- Statistical language model (SLM): Markov assumption, *n*-gram language models
- Neural language model (NLM): characterize the probability of word sequences by neural networks, word2vec
- Pre-trained language model (PLM): pre-training and fine-tuning, ELMo, BERT(330M)
- Large language model (LLM): large-sized PLMs, GPT-2(1.5B)

Three major differences between LLMs and PLMs:

- LLMs display some surprising *emergent abilities* [40] that may not be observed in previous smaller PLMs.
- LLMs would revolutionize the way that humans develop and use AI algorithms.
- The development of LLMs no longer draws a clear distinction between research and engineering.

## Scaling Laws for LLMs I

*Large language models* (LLMs) refer to Transformer language models that contain tens of or hundreds of billions (or more) of parameters. ($\geq$10B)

Extensive research has shown that *scaling* can largely improve the model capacity of LLMs.

- KM scaling law. (Open AI)

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13}$$
$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13}$$
$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^{8}$$

$N$: model size, $D$: dataset size, $C$: the amount of training compute, $L(\cdot)$: the cross entropy loss, $c$: compute budget.

## Scaling Laws for LLMs II

- Chinchilla scaling law. (Google DeepMind)

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

  where $E = 1.69, A = 406, B = 410.7, \alpha = 0.34, \beta = 0.28$.

The KM scaling law favors a larger budget allocation in *model size* than the data size, while the Chinchilla scaling law argues that the two sizes should be increased in *equal scales*,

## Emergent Abilities I

Emergent abilities of LLMs are the abilities that are not present in small models but arise in large models [40].

- In-context learning (ICL). It's formally introduced by 175B GPT-3 [2].
  Assuming that the language model has been provided with a natural language instruction and/or several task demonstrations, it can generate the expected output for the test instances by completing the word sequence of input text, *without requiring additional training or gradient update*.
- Instruction following.
  By fine-tuning with a mixture of multi-task datasets formatted via natural language descriptions (called *instruction tuning*), LLMs are shown to perform well on unseen tasks that are also described in the form of instructions.

## Emergent Abilities II

- Step-by-step reasoning.
  With the *chain-of-thought* (CoT) prompting strategy [41], LLMs can solve such tasks by utilizing the prompting mechanism that involves intermediate reasoning steps for deriving the final answer.

## Key Techniques for LLMs I

- Scaling.
  GPT-3 and PaLM explored the scaling limits by increasing the model size to 175B and 540B, respectively. Since compute budget is usually limited, scaling laws can be further employed to conduct a more compute-efficient allocation of the compute resources.
- Training.
  To support distributed training, several optimization frameworks have been released to facilitate the implementation and deployment of parallel algorithms, such as DeepSpeed [31] and Megatron-LM [36].
- Ability eliciting.
  It is useful to design suitable task instructions or specific in-context learning strategies to elicit potential abilities of LLMs.

## Key Techniques for LLMs II

- Alignment tuning.
  It is necessary to align LLMs with human values, e.g., helpful, honest, and harmless. For this purpose, InstructGPT [29] designs an effective tuning approach that enables LLMs to follow the expected instructions, which utilizes the technique of *reinforcement learning with human feedback* (RLHF).
- Tools manipulation.
  ChatGPT has enabled the mechanism of using external plugins (existing or newly created apps), which are by analogy with the ″eyes and ears″ of LLMs.

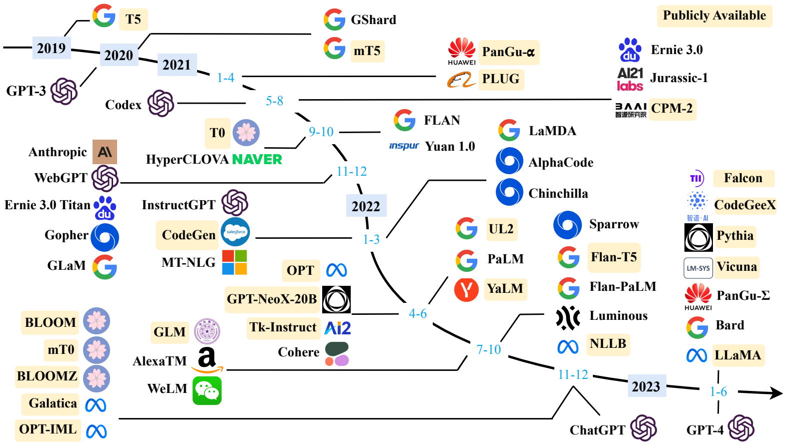


Fig. 1.1: A timeline of existing large language models (having a size larger than 10B) in recent years.

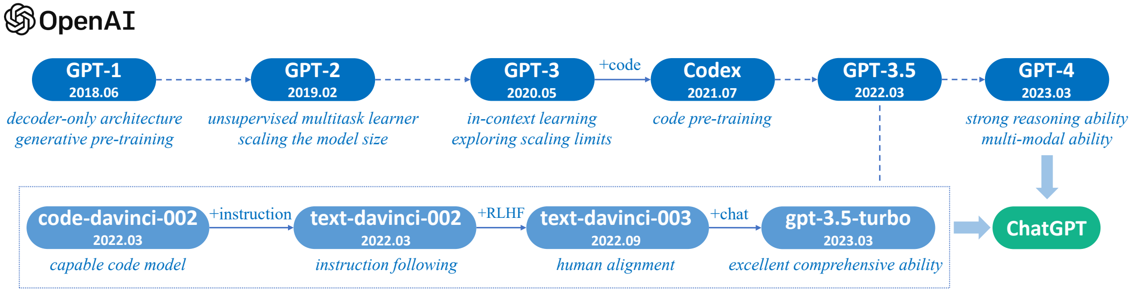


Fig. 1.2: Technical evolution of GPT-series models.

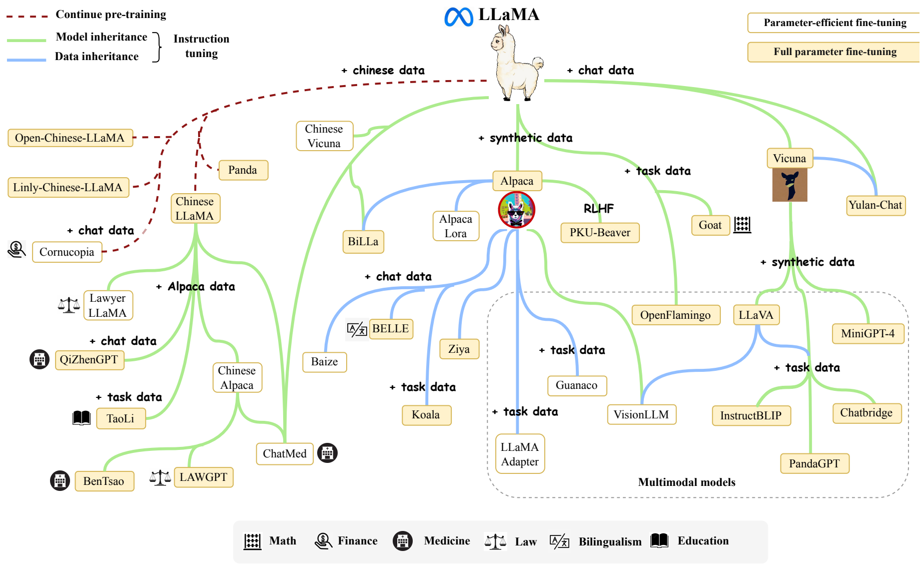


Fig. 1.3: Research work conducted on LLaMA

## Contents

## Data

General Text Data

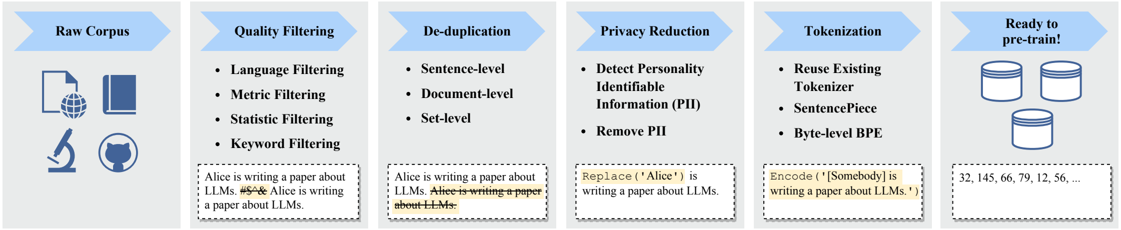
- *Webpages.* LLMs can gain diverse linguistic knowledge and generalization capabilities. CommonCrawl[1] contains a large amount of data from the web.
- *Conversation text* can enhance the conversational competence of LLMs and potentially improve their performance on a range of question-answering tasks.
- *Books* provide an important source of formal long texts.

Specialized Text Data

- *Multilingual text* can enhance the multilingual abilities of language understanding and generation. For example, PaLM [3] have curated multilingual data covering 122 languages.
- *Scientific text* is mainly collected from arXiv papers, scientific textbooks, math webpages, and other related scientific resources.
- *Code*. [12] also speculates that training on code might be a source of complex reasoning abilities (e.g., chain-of-thought ability).

[1] https://commoncrawl.org/

## Data Preprocessing



Raw Corpus
Quality Filtering
• Language Filtering
• Metric Filtering
• Statistic Filtering
• Keyword Filtering
Alice is writing a paper about LLMs. ~~#$^&~~ Alice is writing a paper about LLMs.
De-duplication
• Sentence-level
• Document-level
• Set-level
Alice is writing a paper about LLMs. ~~Alice is writing a paper about LLMs.~~
Privacy Reduction
• Detect Personality Identifiable Information (PII)
• Remove PII
Replace('Alice') is writing a paper about LLMs.
Tokenization
• Reuse Existing Tokenizer
• SentencePiece
• Byte-level BPE
Encode('[Somebody] is writing a paper about LLMs.')
Ready to pre-train!
32, 145, 66, 79, 12, 56, ...

Fig. 1.4: Typical data preprocessing pipeline for pre-training large language models.

## Architecture I

Transformer architecture has become the de facto backbone to develop various LLMs.

- *Encoder-decoder*. The vanilla Transformer model is built on the encoder-decoder architecture. So far, there are only a small number of LLMs that are built based on the encoder-decoder architecture.
- *Causal Decoder.* The causal decoder architecture incorporates the unidirectional attention mask, to guarantee that each input token can only attend to the past tokens and itself. It's also known as " *decoder-only architecture* ".
- *Prefix Decoder.* The prefix decoder architecture revises the masking mechanism of causal decoders, to enable performing bidirectional attention over the prefix tokens and unidirectional attention only on generated tokens. (GLM-130B)

## Architecture II

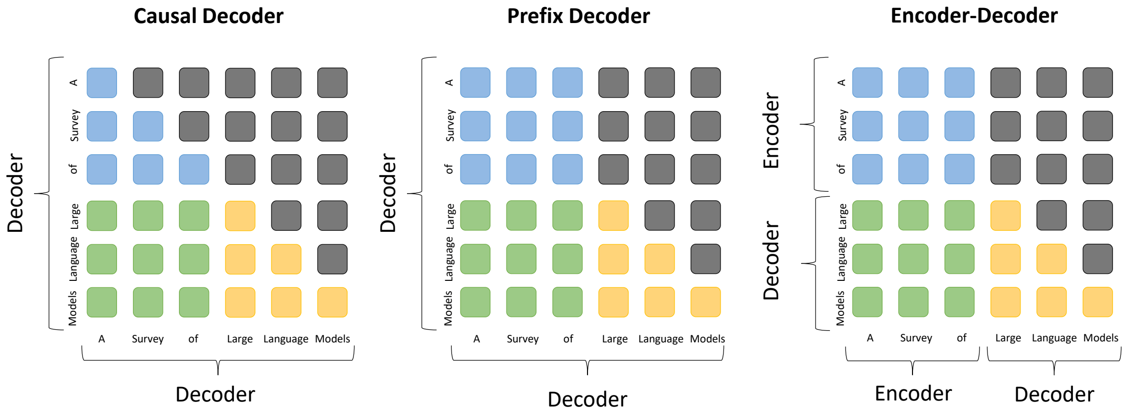


Fig. 1.5: A comparison of the attention patterns in three mainstream architectures.

## Pre-training Tasks

- Language Modeling.
  Given a sequence of tokens $\mathbf{x} = \{x_1, ..., x_n\}$, the LM task aims to autoregressively predict the target tokens $x_i$ based on the preceding tokens $x_{<i}$ in a sequence.

$$\mathcal{L}_{LM}(\mathbf{x}) = \sum_{i=1}^{n} \log P(x_i \mid x_{<i})$$

- Denoising Autoencoding.
  The inputs $\mathbf{x}_{\backslash \tilde{\mathbf{x}}}$ for DAE task are corrupted text with randomly replaced spans. Then, the language models are trained to recover the replaced tokens $\tilde{\mathbf{x}}$.

$$\mathcal{L}_{DAE}(\mathbf{x}) = \log P(\tilde{\mathbf{x}} \mid \mathbf{x}_{\backslash \tilde{\mathbf{x}}})$$

- Mixture-of-Denoisers. (MoD)
  MoD regards both LM and DAE objectives as different types of denoising tasks, namely S-denoiser (LM), R-denoiser (DAE, short span and low corruption), and X-denoiser (DAE, long span or high corruption).

# Contents

## Formatted Instance Construction I

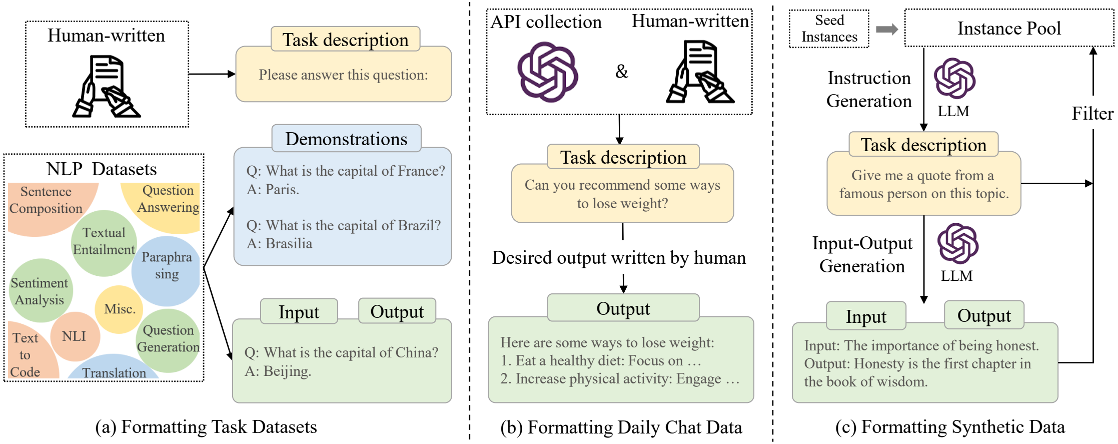
In order to perform instruction tuning, we need to

1. collect or construct instruction-formatted instances
2. employ these formatted instances to fine-tune LLMs in a supervised learning way (e.g., training with the sequence-to-sequence loss).

Three major methods for constructing formatted instances.

1. *Task Datasets.* Collect the instances form a diverse range of tasks (e.g., text summarization, translation).
2. *Daily Chat Data.* InstructGPT proposes to take the queries that real users have submitted to the OpenAI API as the task descriptions. Labelers directly answer these instructions as the output.
3. *Synthetic Data.* Feed existing instances into LLMs to synthesize diverse task descriptions and instances.

## Formatted Instance Construction II



Human-written
Task description
Please answer this question:
NLP Datasets
Sentence Composition
Question Answering
Textual Entailment
Paraphrasing
Sentiment Analysis
Misc.
Text to Code
NLI
Question Generation
Translation
Demonstrations
Q: What is the capital of France?
A: Paris.
Q: What is the capital of Brazil?
A: Brasilia
Input
Output
Q: What is the capital of China?
A: Beijing.
(a) Formatting Task Datasets
API collection
Human-written
&
Task description
Can you recommend some ways to lose weight?
Desired output written by human
Output
Here are some ways to lose weight:
1. Eat a healthy diet: Focus on …
2. Increase physical activity: Engage …
(b) Formatting Daily Chat Data
Seed Instances
Instance Pool
Instruction Generation
LLM
Filter
Task description
Give me a quote from a famous person on this topic.
Input-Output Generation
LLM
Input
Output
Input: The importance of being honest.
Output: Honesty is the first chapter in the book of wisdom.
(c) Formatting Synthetic Data

Fig. 1.6: Three different methods for constructing the instruction-formatted instances.

## Effects of Instruction Tuning

- *Peformance Improvement.* Instruction tuning has become an important way to improve or unlock the abilities of LLMs [4].
- *Task Generalization.* It endows LLMs with the ability (often considered as an emergent ability) to follow human instructions to perform specific tasks without demonstrations, even on unseen tasks.
- *Domain Specialization.* Instruction tuning is an effective approach to adapting existing general LLMs to be domain-specific experts. For instance, researchers propose to fine-tune Flan-PaLM [4] using medical datasets to create Med-PaLM [37], a medical knowledge assistant that achieves performance levels comparable to those of expert clinicians.

## Device Requirements of Instruction Tuning

| Models | A800 Full Training | | | A800 LoRA Training | | | A800 Inference (16-bit) | | 3090 Inference (16-bit) | | 3090 Inference (8-bit) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #GPU | BS | Time | #GPU | BS | Time | #GPU | #Token/s | #GPU | #Token/s | #GPU | #Token/s |
| LLaMA-7B | 2 | 8 | 3.0h | 1 | 80 | 3.5h | 1 | 36.6 | 1 | 24.3 | 1 | 7.5 |
| LLaMA-13B | 4 | 8 | 3.1h | 1 | 48 | 5.1h | 1 | 26.8 | 2 | 9.9 | 1 | 4.5 |
| LLaMA-30B | 8 | 4 | 6.1h | 1 | 24 | 14.3h | 1 | 17.7 | 4 | 3.8 | 2 | 2.6 |
| LLaMA-65B | 16 | 2 | 11.2h | 1 | 4 | 60.6h | 2 | 8.8 | 8 | 2.0 | 4 | 1.5 |

Fig. 1.7: Basic statistics of the required number of GPUs, tuning time, batch size (denoted as BS) per device (full tuning and LoRA tuning), and inference rate (the number of generated tokes per second).

## Contents

## Background and Criteria for Alignment

- LLMs may sometimes exhibit unintended behaviors, e.g., fabricating false information, pursuing inaccurate objectives, and producing harmful, misleading, and biased expressions.
- It has been shown that alignment might harm the general abilities of LLMs to some extent, which is called *alignment tax* in related literature.

Alignment Criteria

- *Helpfulness.* To be helpful, the LLM should demonstrate a clear attempt to assist users in solving their tasks or answering questions in a concise and efficient manner as possible.
- *Honesty.* A LLM aligned to be honest should present accurate content to users instead of fabricating information. Additionally, it is crucial for the LLM to convey appropriate degrees of uncertainty in its output.
- *Harmlessness.* It requires that the language produced by the model should not be offensive or discriminatory.

## Collecting Human Feedback

- *Human Labeler Selection.*
  Researchers first label a small amount of data and then measure the agreement between themselves and human labelers. The labelers with the highest agreement will be selected to proceed with the subsequent annotation work.
- *Human Feedback Collection.*
  - *Ranking-based approach.* Human labelers often evaluate model-generated outputs in a coarse-grained manner (i.e., only selecting the best) without taking into account more fine-grained alignment criteria.
  - *Question-based approach.* Human labelers can provide more detailed feedback by answering certain questions designed by researchers, covering the alignment criteria as well as additional constraints for LLMs.
  - *Rule-based approach.* Many studies also develop rule-based methods to provide more detailed human feedback.

## Reinforcement Learning from Human Feedback I

Reinforcement learning from human feedback (RLHF) has been widely used for alignment tuning in recent powerful LLMs, such as ChatGPT. The RLHF system mainly comprises

1. *a pre-trained LM to be aligned*
   The pre-trained LM is typically a generative model that is initialized with existing pretrained LM parameters. For example, OpenAI uses 175B GPT-3.
2. *a reward model (RM) learning from human feedback*
   The RM provides (learned) guidance signals that reflect human preferences for the text generated by the LM, usually in the form of a scalar value. Existing work typically employs reward models *having a parameter scale different from that of the aligned LM*. For example, OpenAI uses 6B GPT-3 as the reward model.
3. *a RL algorithm training the LM*
   Proximal Policy Optimization (PPO) is widely used.

## Reinforcement Learning from Human Feedback II

Key Steps for RLHF.

- *Supervised fine-tuning.* To make the LM initially perform desired behaviors, it usually needs to collect a supervised dataset containing input prompts (instruction) and desired outputs for fine-tuning the LM.
- *Reward model training.* We employ the LM to generate a certain number of output texts using sampled prompts (from either the supervised dataset or the human-generated prompt) as input. We then invite human labelers to annotate the preference for these pairs. Then, the RM is trained to predict the human-preferred output.
- *RL fine-tuning.* The pre-trained LM acts as the policy that takes as input a prompt and returns an output text, the action space of it is the vocabulary, the state is the currently generated token sequence, and the reward is provided by the RM.

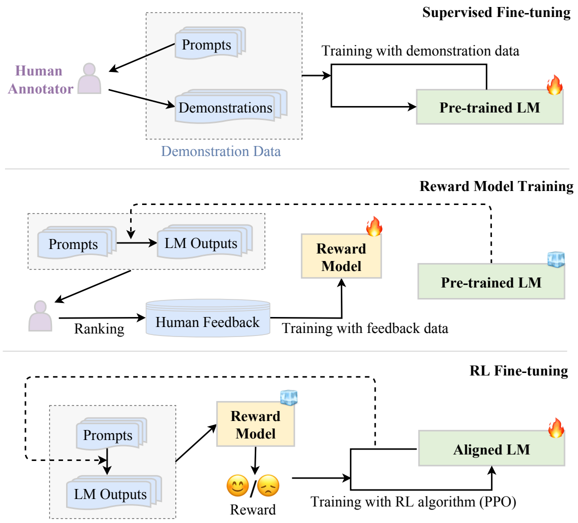


Fig. 1.8: The workflow of the RLHF algorithm.

## Contents

## Parameter-efficient Fine-tuning I

Parameter-efficient fine-tuning (PEFT) aims to reduce the number of trainable parameters while retaining a good performance as possible.

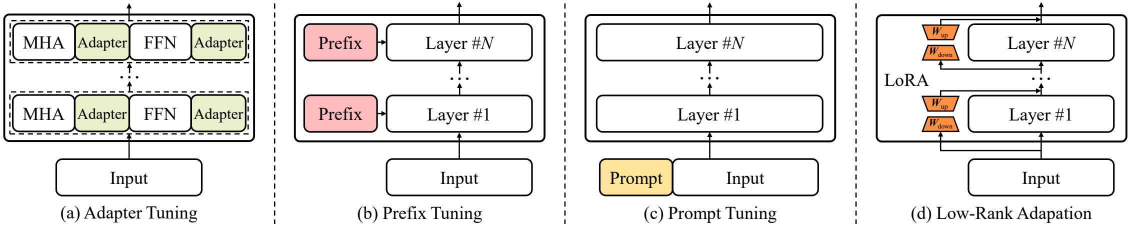


Fig. 1.9: Four different parameter-efficient fine-tuning methods.

## Parameter-efficient Fine-tuning II

- *Adapter tuning* incorporates small neural network modules (called adapter) into the Transformer models [17]. During fine-tuning, the adapter modules would be optimized according to the specific task goals, while the parameters of the original language model are frozen.
- *Prefix tuning* [24] prepends a sequence of prefixes, which are a set of trainable continuous vectors, to each Transformer layer in language models. During fine-tuning, only the prefix parameters would be trained.
- *Prompt tuning* [22] mainly focuses on incorporating trainable prompt vectors at the input layer.

## Parameter-efficient Fine-tuning III

- *Low-Rank Adaptation (LoRA)* [18] imposes the low-rank constraint for approximating the update matrix at each dense layer, so as to reduce the trainable parameters for adapting to downstream tasks.
  Consider the case of optimizing a parameter matrix $\mathbf{W}$ as follows
  $$\mathbf{W} \leftarrow \mathbf{W} + \Delta\mathbf{W}$$
  LoRA freeze the original matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ amd approximate the parameter update $\Delta\mathbf{W}$ by
  $$\Delta\mathbf{W} = \mathbf{A} \cdot \mathbf{B}^\top$$
  where $\mathbf{A} \in \mathbb{R}^{m \times k}, \mathbf{B} \in \mathbb{R}^{n \times k}$ and $k \ll \min(m, n)$.

LoRA has been widely applied to open-source LLMs (e.g., LLaMA)

## Memory-efficient Model Adaptation

We can reduce the memory footprint of LLMs via *model quantization* so that large-sized LLMs can be used in resource-limited settings.

- Quantization often refers to the mapping process from floating-point numbers to integers [15], especially the 8-bit integer quantization (i.e., *INT8 quantization*).

  $$x_q = R(x/S) - Z$$

  which transforms a floating number $x$ into a quantized value $x_q$. $S$ and $Z$ denote the scaling factor and zero-point factor (determining symmetric or asymmetric quantization), and $R(\cdot)$ denotes the rounding operation.

## Quantization Methods for LLMs I

*Quantization-aware training (QAT)* (requiring additional full model retraining)
*Post-training quantization (PTQ)* (requires no model retraining)

- *Mixed-precision decomposition.* Recover the outlier in a high precision.
- *Fine-grained quantization.* Apply different quantization approach for activations and weights.
- *Balancing the quantization difficulty.* Migrate the difficulty from activations to weights.
- *Layer-wise quantization.* Find optimal quantized weights that minimize a layer-wise reconstruction loss.

Empirical Findings.

- INT8 weight quantization can often yield very good results on LLMs, while the performance of lower precision weight quantization depends on specific methods.
- Activations are more difficult to be quantized than weights.

## Quantization Methods for LLMs II

- Efficient fine-tuning enhanced quantization is a good option to enhance the performance of quantized LLMs.
  QLoRA [7] incorporates additional small tunable adapters (16-bit precision) into the quantized models, to achieve an efficient, high-precision model fine-tuning.

## Contents

## In-context Learning I

As a special prompting form, in-context learning (ICL) is first proposed along with GPT-3 [2], which has become a typical approach to *utilizing* LLMs.

ICL uses a formatted natural language prompt, consisting of the *task description* and/or a few *task examples* as demonstrations.

- Let $D_k = \{f(x_1, y_1), \ldots, f(x_k, y_k)\}$ represent a set of demonstrations with $k$ examples, where $f(x_k, y_k)$ is the prompt function that transforms the $k$-th task example into natural language prompts.
- Given the task description $I$, demonstration $D_k$, and a new input query $x_{k+1}$, the prediction of the output $\hat{y}_{k+1}$ is

$$\mathrm{LLM}(I, \underbrace{f(x_1, y_1), \ldots, f(x_k, y_k)}_{\text{demonstrations}}, f(\underbrace{x_{k+1}}_{\text{input}}, \underbrace{\_\_\_}_{\text{answer}})) \rightarrow \hat{y}_{k+1}$$

## In-context Learning II

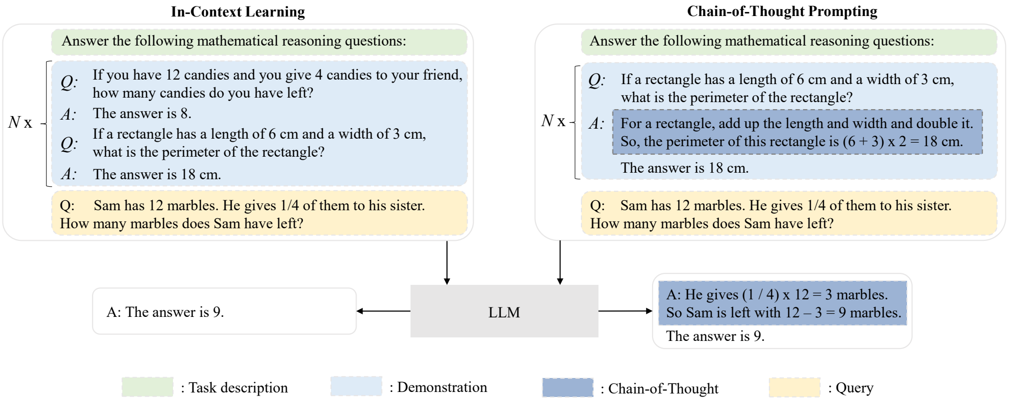

In-Context Learning
Answer the following mathematical reasoning questions:
N x
Q: If you have 12 candies and you give 4 candies to your friend, how many candies do you have left?
A: The answer is 8.
Q: If a rectangle has a length of 6 cm and a width of 3 cm, what is the perimeter of the rectangle?
A: The answer is 18 cm.
Q: Sam has 12 marbles. He gives 1/4 of them to his sister. How many marbles does Sam have left?
Chain-of-Thought Prompting
Answer the following mathematical reasoning questions:
N x
Q: If a rectangle has a length of 6 cm and a width of 3 cm, what is the perimeter of the rectangle?
A: For a rectangle, add up the length and width and double it. So, the perimeter of this rectangle is (6 + 3) x 2 = 18 cm.
The answer is 18 cm.
Q: Sam has 12 marbles. He gives 1/4 of them to his sister. How many marbles does Sam have left?
LLM
A: The answer is 9.
A: He gives (1 / 4) x 12 = 3 marbles. So Sam is left with 12 – 3 = 9 marbles.
The answer is 9.
: Task description
: Demonstration
: Chain-of-Thought
: Query

Fig. 1.10: A comparative illustration of in-context learning (ICL) and chain-of-thought (CoT) prompting.

## In-context Learning III

The performance of ICL heavily relies on demonstrations

- Demonstration Selection.
  - *Heuristic approaches.* Several studies employ a $k$-NN based retriever to select examples that are semantically relevant to the query.
  - *LLM-based approaches.* LLMs can be utilized to directly measure the informativeness of each example according to the performance gain after adding the example.
- Demonstration Format. A straightforward method is to instantiate a pre-defined template with the corresponding input-output pairs.
- Demonstration Order. Demonstrations can be directly organized according to their similarity to the query in the embedding space [27]: the more similar, the closer to the end.

## Chain-of-thought Prompting

Chain-of-Thought (CoT) [41] is an improved prompting strategy to boost the performance of LLMs on complex reasoning tasks, such as arithmetic reasoning. CoT incorporates *intermediate reasoning steps* that can lead to the final output into the prompts.

- *Few-shot CoT* is a special case of ICL, which augments each demonstration by incorporating the CoT reasoning steps.
- *Zero-shot CoT* does not include human-annotated task demonstrations in the prompts. Instead, it directly generates reasoning steps and then employs the generated CoTs to derive the answers.
  Zero-shot CoT is first proposed in [21], where the LLM is first prompted by *"Let's think step by step"* to generate reasoning steps and then prompted by *"Therefore, the answer is"* to derive the final answer.

## Contents

## Overall Framework I

Prompt-based planning has been proposed to break down complex tasks into smaller subtasks and generate a plan of actions to accomplish the task.

- *Task planner*, which is played by LLMs, aims to generate the whole plan to solve a target task.
- *Plan executor* is responsible for executing the actions in the plan. It can be implemented by models like LLMs for textual tasks or by objects like robots for embodied tasks.
- *Environment* refers to where the plan executor carries out the actions, which can be set differently according to specific tasks, e.g., the LLM itself or external virtual world like Minecraft.

# Overall Framework II

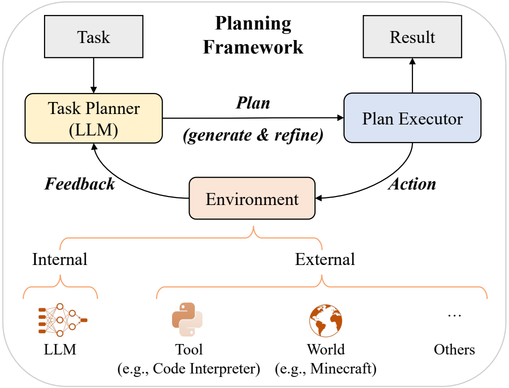


Fig. 1.11: An illustration of the formulation for prompt based planning by LLMs for solving complex tasks.

## Key Steps I

1. *Plan generation* focuses on directly generating action sequences by prompting LLMs.
   - Text-based approaches generate plans in the form of natural language, e.g., ToolFormer [32].
   - Code-based approaches generate more verifiable plans in the form of executable code in programming languages (Python), e.g., LLM+P [25].
2. *Feedback acquisition*.
   - Internal feedback. The LLM itself can be utilized as a feedback provider.
     Reflexion [35] utilizes LLMs to transform sparse result signals (e.g., success or failure) into concrete text-based feedback (e.g., *"You should recommend comedies that the user mentions in the query instead of horror movies"*) and stores this feedback in long-term memory for future planning.
   - External feedback. For example, code interpreters, virtual worlds like Minecraft.

## Key Steps II

3. *Plan Refinement*.
   - Reasoning.
     The feedback data from the environment may not be directly suitable to be utilized by LLMs for plan refinement. Some work adds the explicit reasoning process to extract critical information from feedback.
   - Backtracking.
     Early methods mainly consider planning forward actions while maintaining the existing plan, thus likely leading to *local optimal plans* based on a short-term evaluation. To solve this, Tree of Thoughts [47] allows backtracking with search algorithms like breadth-first and depth-first search to make global planning.
   - Memorization.
     In order to handle long-horizon tasks, it has become a key approach to aid plan refinement with long-term memory.

## Contents

## Basic Ability

Three basic types of ability evaluation for LLMs:

1. Language Generation
2. Knowledge Utilization
3. Complex Reasoning

## Language Generation I

- *Language Modeling* aims to predict the next token based on the previous token.
  Datasets: Penn Treebank, WikiText-103, Pile and LAMBADA (to predict the last word of sentences based on a paragraph of context).
  Metric: accuracy and perplexity.
- *Conditional Text Generation* focuses on generating texts satisfying specific task demands based on the given conditions, typically including machine translation, text summarization, question answering, and strutured data generation and long text generation for LLMs.
  Metric: accuracy, BLEU, ROUGE, and human ratings.
- *Code Synthesis* means generating formal language. LLMs show strong abilities to generate computer programs.
  Metric: Pass rate (checked by compilers or interpreters)

## Language Generation II

Two major issues:

- *Unreliable generation evaluation*: LLMs have been capable of generating texts with a comparable quality to human-written texts, which however might be underestimated by automatic reference-based metrics.
  - Inconsistency between human evaluation and automatic reference-based metrics
  - Difficult to achieve consensus among human annotators
- *Underperforming specialized generation*: LLMs may fall short in mastering generation tasks that require domain-specific knowledge or generating structured data.

## Knowledge Utilization I

Knowledge utilization is an important ability of intelligent systems to accomplish knowledge-intensive tasks (e.g., commonsense question answering and fact completion) based on supporting factual evidence.

- *Closed-Book QA* tests the acquired factual knowledge of LLMs from the pre-training corpus, where LLMs should answer the question only based on the given context without using external resources.
  Datasets: Natural Questions, Web Questions, and TriviaQA.
  Metric: Accuracy.
- *Open-Book QA* allows LLMs to extract useful evidence from the external knowledge base or document collections, and then answer the question based on the extracted evidence. LLMs for Open-Book QA tasks are often paired with a text retriever (or even a search engine).
  Datasets: have overlap with close-book QA datasets, but with external data sources.

## Knowledge Utilization II

- *Knowledge Completion*. LLMs might be considered as a knowledge base, which can be leveraged to complete or predict the missing parts of knowledge units (e.g., knowledge triples).
  Datasets: Knowledge graph completion tasks (FB15k-237) and fact completion tasks (WikiFact).
  It is difficult for existing LLMs to accomplish knowledge completion tasks related to *specific relation types*.

Major Issues.

- *Hallucination.* The generated information is either in conflict with the existing source (intrinsic hallucination) or cannot be verified by the available source (extrinsic hallucination)
  1. Existing work shows that LLMs encounter difficulties in recognizing the hallucinated content in text.
  2. LLMs still lack an ability to accurately control the use of internal or external knowledge.

## Knowledge Utilization III

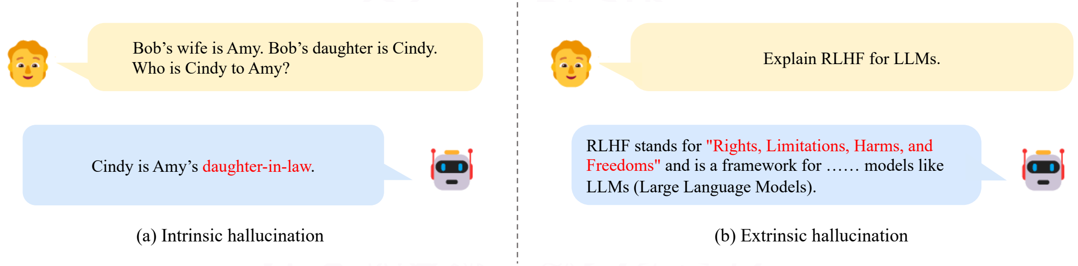

Bob's wife is Amy. Bob's daughter is Cindy. Who is Cindy to Amy?
Cindy is Amy's daughter-in-law.
(a) Intrinsic hallucination
Explain RLHF for LLMs.
RLHF stands for "Rights, Limitations, Harms, and Freedoms" and is a framework for …… models like LLMs (Large Language Models).
(b) Extrinsic hallucination

Fig. 1.12: Examples of intrinsic and extrinsic hallucination for a public LLM.

- *Knowledge Recency.* LLMs would encounter difficulties when solving tasks that require the latest knowledge beyond the training data.
  It is very costly to fine-tune LLMs, and also likely to cause the catastrophic forgetting issue when incrementally training LLMs.

## Complex Reasoning I

Complex reasoning refers to the ability of understanding and utilizing supporting evidence or logic to derive conclusions or make decisions.

- *Knowledge Reasoning.* The knowledge reasoning tasks rely on logical relations and evidence about factual knowledge to answer the given question.
  Dataset: CSQA, StrategyQA (commonsense reasoning) and ScienceQA (science knowledge reasoning).
  Chian-of-thought (CoT) prompting strategy improves the reasoning performance.
- *Symbolic Reasoning* mainly focuses on manipulating the symbols in a formal rule setting to fulfill some specific goal, where the operations and rules may have never been seen by LLMs during pretraining.
  Evaluation task: Last letter concatenation.

## Complex Reasoning II

- *Mathmetical Reasoning* need to comprehensively utilize mathematical knowledge, logic, and computation for solving problems or generating proof statements.
  1. Math problem solving tasks
     Dataset: SVAMP, GSM8k, and MATH
     Strategy: CoT prompting and continually pre-training
  2. Automated theorem proving (ATP)
     Dataset: PISA and miniF2F

Major Issues

- *Reasoning inconsistency.* LLMs may generate the correct answer following an invalid reasoning path, or produce a wrong answer after a correct reasoning process, leading to inconsistency between the derived answer and the reasoning process.
  Solution: Fine-tune LLMs with process-based feedback.

## Complex Reasoning III

- *Numerical computation.* LLMs still face difficulties in the involved numerical computation, especially for the symbols that are seldom encountered during pre-training, such as arithmetic with large numbers.
  Solution: External tools or tokenize digits into individual tokens.

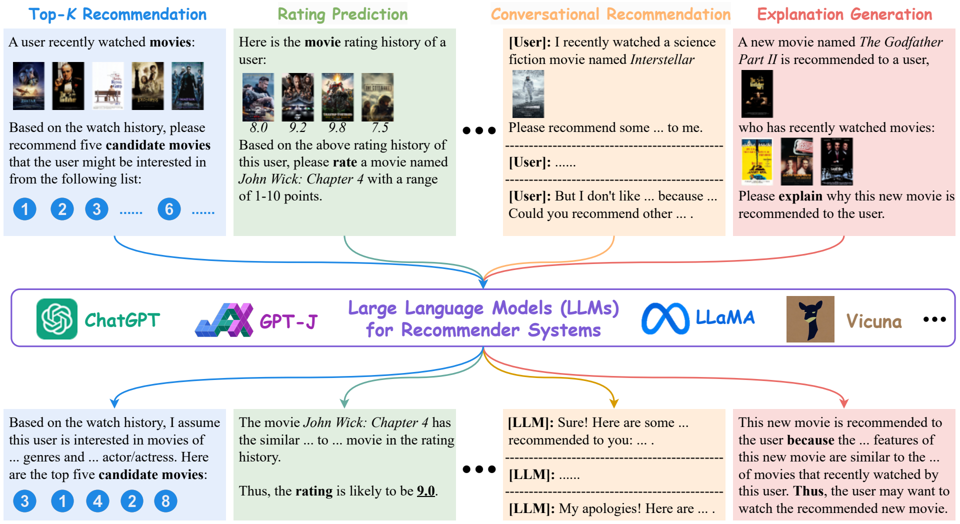
## Advanced Ability I

- *Human Alignment.* It is desired that LLMs could well conform to human values and needs.
  1. Helpfulness and honesty: adversarial question answering tasks.
  2. Harmlessness: CrowS-Pairs and Winogender.
- *Interaction with External Environment.* LLMs have the ability to receive feedback from the external environment and perform actions according to the behavior instruction, e.g., generating action plans in natural language to manipulate agents.
  1. Several embodied AI environments: VirtualHome, ALFRED, BEHAVIOR, Apart, ect.
  2. Multi-agent collaboration.
- *Tool Manipulation.* When solving complex problems, LLMs can turn to external tools if they determine it is necessary, e.g., seach engine, calculator, and compiler.
  Math problem solving and knowledge question answering.

# Contents

## Limitations of Current RecSys

1. Previous DNN-based models (e.g., CNN and LSTM) and pre-trained language models (e.g., BERT) for recommender systems cannot *sufficiently capture textual knowledge* about users and items, demonstrating their inferior natural language understanding capability.
2. Most existing RecSys methods have been specifically designed for their own tasks and have inadequate *generalization ability* to their unseen recommendation tasks.
3. Most existing DNN-based recommendation methods can achieve promising performance on recommendation tasks needing simple decisions (e.g., rating prediction, and top-$k$ recommendations). However, they face difficulties in supporting *complex and multi-step decisions* that involve multiple reasoning steps (e.g., trip planning recommendations).



A user recently watched **movies**:

Based on the watch history, please recommend five **candidate movies** that the user might be interested in from the following list:

Here is the **movie** rating history of a user:

Based on the above rating history of this user, please **rate** a movie named *John Wick: Chapter 4* with a range of 1-10 points.

**[User]:** I recently watched a science fiction movie named *Interstellar*

Please recommend some ... to me.

**[User]:** ......

**[User]:** But I don't like ... because ... Could you recommend other ... .

A new movie named *The Godfather Part II* is recommended to a user,

who has recently watched movies:

Please **explain** why this new movie is recommended to the user.

Based on the watch history, I assume this user is interested in movies of ... genres and ... actor/actress. Here are the top five **candidate movies**:

The movie *John Wick: Chapter 4* has the similar ... to ... movie in the rating history.

Thus, the **rating** is likely to be **9.0**.

**[LLM]:** Sure! Here are some ... recommended to you: ... .

**[LLM]:** ......

**[LLM]:** My apologies! Here are ... .

This new movie is recommended to the user **because** the ... features of this new movie are similar to the ... of movies that recently watched by this user. **Thus**, the user may want to watch the recommended new movie.

Fig. 2.1: Examples of the applications of LLMs for various recommendation tasks in the scenario of movie recommendations.

## LLM-empowered RecSys

Two types of LLM-empowered RecSys that take advantage of LLMs to learn the *representation* of users and items

1. ID-based Recommender Systems
   Modern recommendation approaches are proposed to model these behaviors by learning embedding vectors of each ID representation.
2. Textual Side Information-enhanced Recommender Systems
   - Pure ID indexing of users and items is naturally discrete, which cannot provide sufficient semantic information.
   - It is very challenging to perform relevance calculations based on index representations.
   - ID indexing usually requires modifying the vocabularies and altering the parameters of LLMs, which brings additional computation costs.

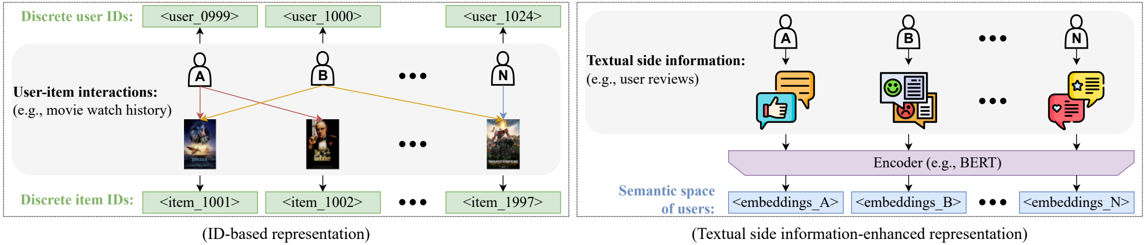


Fig. 2.2: An illustration of two methods for representing users and items for LLM-based RecSys.

## ID-based Recommender Systems

As the early exploration of LLM-based methods, a unified paradigm called P5 [14] is proposed.

- It transfer various recommendation data formats, such as user-item interactions, user profiles, item descriptions, and user reviews, into natural language sequences by mapping users and items into indexes.
- The pre-trained T5 backbone is used to train the P5 with *personalized prompts*.
- P5 incorporates the normal index phrase with a pair of angle brackets to treat these indexes as special tokens in the vocabulary of LLMs (e.g., $< item\_6637 >$), avoiding tokenizing the phrases into separate tokens.

## Textual Side Information-enhanced Recommender Systems

- Given the textual side information of an item or a user, language models like BERT can serve as the *text encoder* to map the item or user into the semantic space.
- However, solely relying on language models to encode item descriptions might excessively emphasize text features.
- Zero-Shot Item-based Recommendation (ZSIR) [9] introduces a *Product Knowledge Graph (PKG)* to LLMs to refine item features. User and item embeddings are learned via multiple pre-training tasks upon the PKG.
- ShopperBERT [34] investigates modeling user behaviors to denote user representations, which pre-trains user embedding through several pre-training tasks based on user purchase history.

## Pre-training for Recommender Systems I

Two main pre-training methods:

1. *Masked Language Modeling (MLM)* randomly masks tokens or spans in the sequence and requires LLMs to generate the masked tokens or spans based on the remaining context. (encoder-decoder)
2. *Next Token Prediction (NTP)* requires prediction for the next token based on the given context. (decoder-only)

## Pre-training for Recommender Systems II

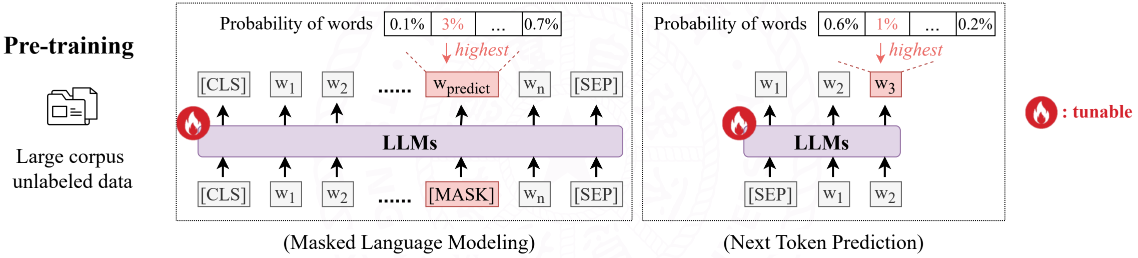


Fig. 2.3: An illustration of two main pre-training methods of LLMs.

## Pre-training for Recommender Systems III

PTUM [42] proposes two similar pre-training tasks

1. *Masked Behavior Prediction (MBP)* masks a single user behavior with the goal of predicting the masked behavior based on the other behaviors in the interaction sequence of the target user.
2. *Next K Behavior Prediction (NBP)* predicts the next $K$ behaviors based on the user-item interaction history.

M6-rec [5] also adopts two pre-training objectives

1. *Text-infilling*
2. *Auto-regressive language generation*

## Fine-tuning for Recommender Systems I

LLMs require fine-tuning to grasp more domain knowledge, which involves training the pre-trained model based on task-specific recommendation datasets that include user-item *interaction behaviors* (e.g., purchase, click, ratings) and *side knowledge* (e.g., users' social relations and items' descriptions).

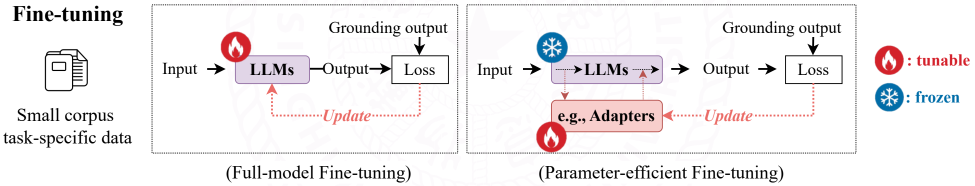


Fig. 2.4: An illustration of two main fine-tuning methods of LLMs.

## Fine-tuning for Recommender Systems II

1. *Full-model Fine-tuning*
   - RecLLM [11] fine-tunes LaMDA as a Conversational Recommender System (CRS) for YouTube video recommendation.
   - GIRL [52] leverages a supervised fine-tuning strategy for instructing LLMs in job recommendation.

   Directly fine-tuning LLMs might bring unintended bias into recommender systems, producing serious harm towards specific groups or individuals based on sensitive attributes such as gender, race and occupation. (LMRec [33])
2. *Parameter-efficient Fine-tuning (PEFT)* involves fine-tuning a small proportion of model weights or a few extra trainable weights.
   - TALLRec [1] introduces an efficient and effective tuning framework on the LLaMA-7B.
   - GLRec [43] takes the advantage of LoRA for fine-tuning and adapting LLMs as job recommender.

## Prompting LLMs for Recommender Systems I

Prompting enables LLMs to unify different downstream tasks into language generation tasks, which are aligned to their objectives during pre-training.

1. *Prompting* keeps LLMs frozen (i.e., no parameters updates), and adapt LLMs to downstream tasks via task-specific prompts.
2. *Prompt Tuning* serves as an additive technique of prompting, which adds new prompt tokens to LLMs and optimizes the prompt based on the task-specific dataset.
3. *Instruction Tuning* trains LLMs to follow prompts as task instructions, rather than to solve specific downstream tasks.

## Prompting LLMs for Recommender Systems II

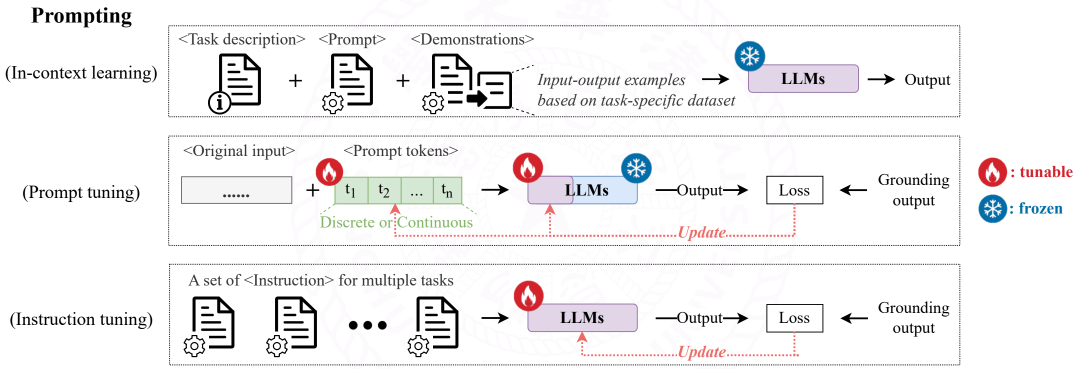


Fig. 2.5: Three representative methods of prompting LLMs.

## Prompting

1. *Conventional Prompting*
   Liu et al. [26] prompt ChatGPT to format the review summary task in recommendations into text summarization, with a prompt including *"Write a short sentence to summarize _."*.
2. *In-Context Learning*
   - Few-shot ICL: Chat-Rec [13].
   - Zero-shot ICL relieves the requirement of taskspecific recommendation datasets to form in-context demonstrations.
     Wang et al. [39] prompt ChatGPT for conversational recommendations with a zero-shot ICL template containing two parts: a text description of conversational recommendation tasks (e.g., *"Recommend items based on user queries in the dialogue."*), and a format guideline in natural languages, such as *"The output format should be <no. > <item title >."*.
3. *Chain-of-Thought Prompting*
   A simple CoT template *"Please infer the preference of the user and recommend suitable items."* is proposed to guide LLMs to first infer the user's explicit preference and then generate final recommendations [50].

## Prompt Tuning

1. *Hard Prompt Tuning* is to generate and update discrete text templates of prompt (e.g., in natural language), for prompting LLMs to specific downstream tasks. (Discrete Optimization)
2. *Soft Prompt Tuning* employs continuous vectors as prompt (e.g., text embeddings), and optimizes the prompt based on task-specific datasets. During soft prompt tuning, only the *soft prompt and minimal parameters at the input layer* of LLMs will be updated.
    - Wu et al. [44] apply contrastive learning to capture user representations and encode them into prompt tokens.
    - Compared to the hard prompt, the soft prompt is more feasible for tuning on continuous space but in a cost of explainability.

## Instruction Tuning

Instruction tuning can be divided into two stages

1. *Instruction Generation Stage.*
   Zhang et al. [50] propose a recommendation-oriented instruction template, including user preferences, intentions, and task forms.
2. *Model Tuning Stage.* Full-model / parameter-efficient tuning.
   TALLRec [1] utilizes LoRA to make the instruction tuning of LLaMA more lightweight.

## Future Directions I

1. Hallucination Mitigation.
   High-stakes scenarios such as medical and legal.
   Employing *factual knowledge graphs* as supplementary factual knowledge during the training and inference stages of LLMs for RecSys is promising to mitigate the hallucination problem.
2. Trustworthy LLMs for RecSys.
   - *Safety & Robustness.* It is crucial to ensure that the output of LLMs for recommender systems is stable given small changes in the LLMs' input.
     Solution: safety-related prompts during RLHF, adversarial training.
   - *Non-discrimination& Fairness.* FaiRLLM [49] and UP5 [19] explore the fairness problem in recommender systems brought by LLMs, which focus on user-side and item generation task.
   - *Explainability.* LLMs for RecSys can be treated as the *'black box'*, complicating the process for users trying to comprehend why a specific output or recommendation was produced.
   - *Privacy.* LLM-based recommender systems often handle sensitive user data, including personal preferences, online behaviors, and other identifiable information.

## Future Directions II

3. Vertical Domain-Specific LLMs for Recommender Systems.
   Medical care [48, 45], law [28], and finance [46].
4. Users & Items Indexing.
   Recent research suggests that LLMs may not perform well when dealing with *long texts* in RecSys, as it can be difficult to effectively capture user-item interaction information in long texts. Therefore, rather than merely using text formats to represent users and items, advanced methods for indexing users & items are desired.
5. Fine-tuning Efficiency.
   The exploration of adapter tuning effects for multi-modal (i.e., both text and image) RecSys is a potential future direction.

## Future Directions III

6. Data Augmentation.
   Utilize LLMs for data augmentation to booster recommendations.
   RecAgnet [38] is a simulation paradigm for recommender systems based on LLMs, which includes a user module for browsing and communication on the social media, and a recommender module for providing search or recommendation lists.

# Contents

## Generative Recommendation I

Vagueness may also be a problem in recommendation scenarios that require precise and unique identifiers of items.

**Definition 2.1 (ID in Recommender Systems)**

*An ID in recommender systems is a sequence of tokens that can uniquely identify an entity, such as a user or an item. An ID can take various forms, such as a vector embedding, a sequence of numerical tokens, and a sequence of word tokens (including an item title, a description of the item, or even a complete news article), as long as it can uniquely identify the entity.*

- For example, a product in e-commerce platform may be assigned the ID "item 7391" and be further represented as a sequence of tokens such as < item><_><73><91> [14].
- IDs resemble token sequences as in text, and thus naturally fit natural language environment as well as LLM.

## Generative Recommendation II

- Due to the huge number of items in real-world systems, traditional RS usually take the *multi-stage filtering* paradigm.
- Advanced recommendation algorithms are not applied to all items, but only a few hundred of items.
- An LLM itself can be the single and entire recommendation pipeline, which directly generates the items to recommend.

**Definition 2.2 (Generative Recommendation)**

*A generative recommender system directly generates recommendations or recommendation-related content without the need to calculate each candidate's ranking score one by one for sorting and ranking.*

Discriminative v.s. Generative

# Generative Recommendation III

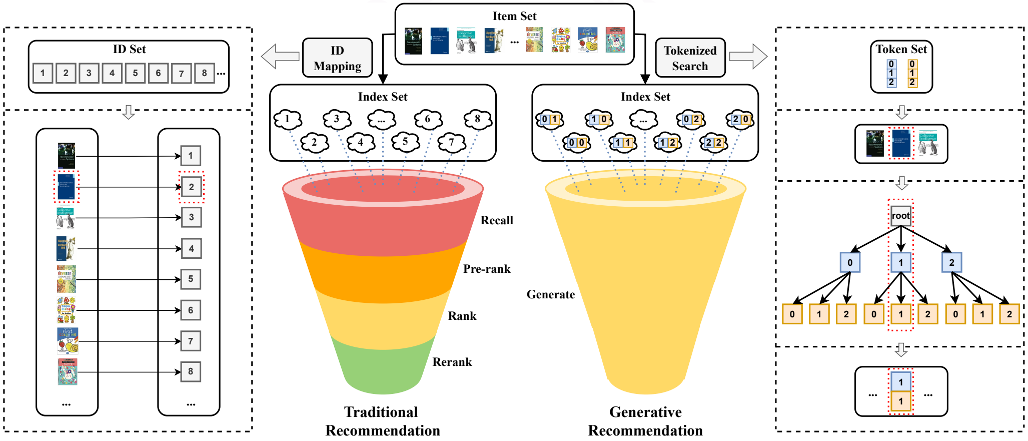


Fig. 2.6: Pipeline comparison between traditional recommender systems and LLM-based generative recommendation.

## Why Generative Recommendation

Problems with discriminative recommendation.

- When the amount of items on a recommendation platform is prohibitively large, the ranking score calculation with regard to each item would be *computationally expensive*.
- Only in the final stage can the relatively complex and advanced models be utilized. Although recent recommendation models are growing more fancy and sophisticated, few have been practically employed in industry.

For generative recommendation.

- At each step of recommendation generation, the LLM can *produce a vector that represents the probability distribution on all possible ID tokens*.
- After a few steps, the generated tokens can constitute a complete ID that stands for the target item.
- We can use finite tokens to represent (almost) infinite items.

  1000 tokens for representing user or item IDs. Each ID consists of 10 tokens. Then we can use these 1000 tokens to represent as many as $1000^{10} = 10^{30}$ items.

## ID Creation Methods I

Intuitively, one would consider the metadata of users and items as an alternative, such as user name and item title.

1. When the IDs are extremely long, e.g., in the case of item description, it would be computationally expensive to conduct generation.
2. It would be difficult to find an exact match in the database for a long ID.
3. Double-checking the existence of each ID would take us back to discriminative recommendation since we need to compare it with each item in the database.
4. Although natural language is a powerful and expressive medium, it can also be vague in many cases.

## ID Creation Methods II

Three typical ID creation approaches

1. *Singular Value Decomposition.* [30]
   - Perform truncated singular value decomposition on user-item interaction data to obtain the item embedding matrix.
   - After a set of operations, including normalization, noise-adding, quantization, and offset adjustment, each item's embedding becomes an array of integers.
2. *Product Quantization.* [16]
   - There are in total $D$ vector sets and each set is comprised of $M$ centroid embeddings.
   - They first encode an item's textual description with BERT to get the item's embedding vector, which is further divided into $D$ segments for quantization.
   - For the $i$-th embedding segment, its nearest centroid embedding from the i-th vector set can be easily found. The index of this centroid embedding then becomes the item's $i$-th ID token.
3. *Collaborative Indexing* [20] compose an item ID with nodes on a hierarchical tree. (Laplacian matrix, spectral clustering)

## How to Do Generative Recommendation I

Construct a prompt template that describes the task and then fill the user and item information such as their IDs in the prompt.
During the inference stage, all kinds of output (e.g., predicted item IDs) are auto-regressively generated as natural language generation.

1. *Rating Prediction*
   - Given a user $u$ and an item $i$, a recommendation model $f(u, i)$ needs to estimate a score $\hat{r}_{u,i}$.
   - $u$ and $i$ are two sequences of tokens. The two IDs can be filled in an instruction prompt $p(u, i)$, e.g., "how would *user*_1234 rate *item*_5678".
2. *Top-N Recommendation*
   Due to the context length limit of LLM, it is impossible to feed the model all the items.
   - Straightforward recommendation uses a prompt that *only contains user information* (ID or user metadata) and asks the LLM to directly generate recommendations for this user [14].
   - Selective Recommendation provides both user information and a list of candidate items in the prompt and asks the LLM to select items for recommendation out of the candidates [30].

## How to Do Generative Recommendation II

3. *Sequential Recommendation*

   We fill the user and the item sequence in a prompt, e.g., "given *user*_1234's interaction history *item*_3456, ..., *item*_4567, *item*_5678, predict the next item with which the user will interact", and then prompt LLM to generate an item ID as a prediction, e.g., "6789".

4. *Explainable Recommendation*
   - A typical LLM-based recommendation explanation task can be natural language explanation generation.
   - Using IDs alone in the prompt could be unclear as to which aspects the model should discuss in the explanation.
   - We can provide some item features f as hint words in the prompt.

5. *Review Generation* is similar to explanation generation, except that reviews are generally longer.

6. *Review Summarization.* It may be unnecessary to summarize a user's own review.

## How to Do Generative Recommendation III

7. *Conversational Recommendation.*

   Although ChatGPT's chatting ability is undeniably impressive, its performance on existing metrics is not very good because they overly stress the matching between generated responses and annotated recommendations or utterances.

8. *Evaluation Protocols*: RMSE, MAE, NDCG, BLEU, ROUGE, etc. More advanced and standard metrics need to be developed.

## Challenges and Opportunities

1. *Hallucination*
2. *Bias and Fairness*
3. *Transparency and Explainability* (Dive into the model and try to explain the internal working mechanism of LLM.)
4. *Controllability*: Users may want to control the features of items or explanations.
5. *Inference Efficiency*
6. *Multimodal Recommendation*: in addition to images, videos and audios can also be generated in an auto-regressive way.
7. *Cold-start Recommendation*

## Contents

## TALLRec

- $\mathcal{T}$itle: TALLRec: An *Effective and Efficient Tuning* Framework to Align Large Language Model with Recommendation [1]
- $\mathcal{A}$uthor:KEQIN BAO, JIZHI ZHANG, YANG ZHANG, WENJIE WANG, FULI FENG, XIANGNAN HE (USTC)
- $\mathcal{P}$ublished: RecSys 2023
- We propose an efficient and effective Tuning framework for Aligning LLMs with Recommendation, namely TALLRec.

## The Ability of LLMs to Make Recommendations

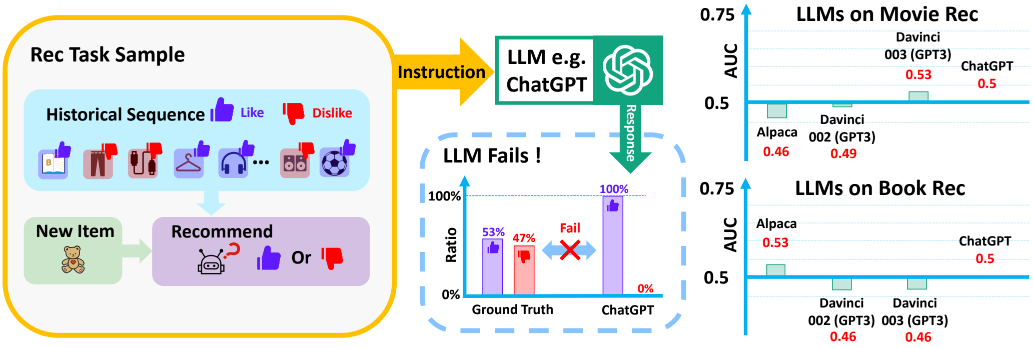


Fig. 2.7: LLMs are given the task of predicting whether a user like the next item based on their interaction history.

Using only In-context Learning may fail to make recommendations.

## TALLRec I

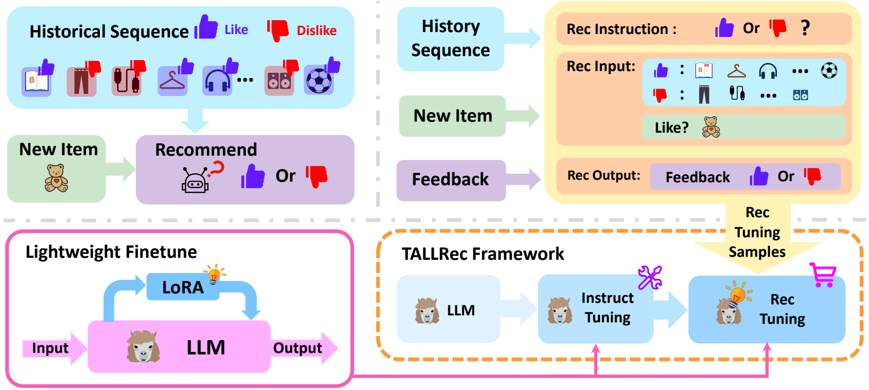


Fig. 2.8: Illustration of the TALLRec framework.

## TALLRec II

TALLRec tuning stages.

1. *Instruction tuning* is the common training process of LLM which enhances LLM's generalization ability. We employ the self-instruct data made available by Alpaca to train our model.
2. *Recommendation tuning (rec-tuning)* emulates the pattern of instruction tuning and fine-tunes the model for the recommendation task.
   Instruction Input: *Rec Instruction* + *Rec Input*
   Instruction Output: *Rec Output*

Backbone: LLaMA-7B.

Lightweight Fine-tuning: LoRA.

## Experiments Settings

Datasets.

- Movie from MovieLens100K.
  Textual descriptions: title and director.
- Book form BookCrossing.
  Textual descriptions: Book-Author and Book-Title.

Few-shot Training Setting: $K$ training samples.

Baselines

- LLM-based methods that use In-context Learning to directly generate recommendations.
- Traditional sequential recommendations.

## Performance Comparison I

| | | Baseline | | | | | | Ours |
|---|---|---|---|---|---|---|---|---|
| | Few-shot | GRU | Caser | SASRec | DROS | GRU-BERT | DROS-BERT | TALLRec |
| movie | 16 | 49.07 | 49.68 | 50.43 | 50.76 | 50.85 | 50.21 | **67.24** |
| | 64 | 49.87 | 51.06 | 50.48 | 51.54 | 51.65 | 51.71 | **67.48** |
| | 256 | 52.89 | 54.20 | 52.25 | 54.07 | 53.44 | 53.94 | **71.98** |
| book | 16 | 48.95 | 49.84 | 49.48 | 49.28 | 50.07 | 50.07 | **56.36** |
| | 64 | 49.64 | 49.72 | 50.06 | 49.13 | 49.64 | 48.98 | **60.39** |
| | 256 | 49.86 | 49.57 | 50.20 | 49.13 | 49.79 | 50.20 | **64.38** |

Fig. 2.9: Performance comparison between conventional sequential recommendation baselines and TALLRec under different few-shot training settings.

## Performance Comparison II

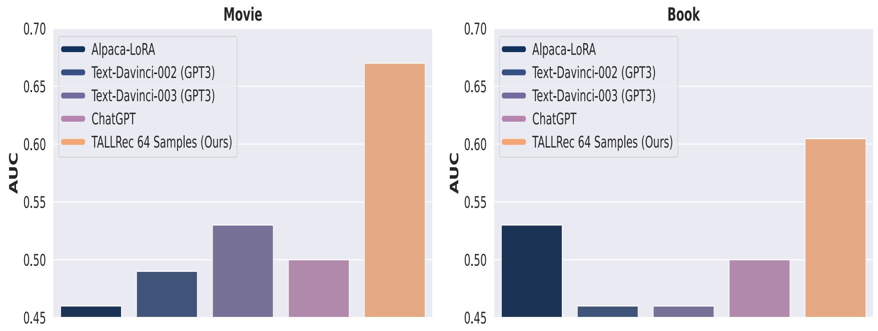


Fig. 2.10: Performance comparison between LLM-based baselines and TALLRec.

## Ablation Study

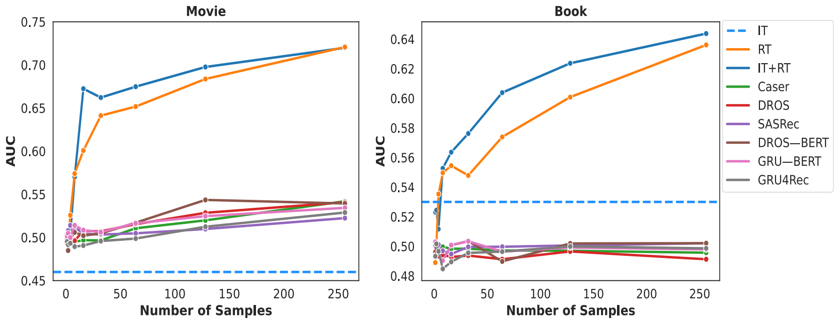


Fig. 2.11: Performance tendency of TALLRec's variants and conventional sequential recommendation methods w.r.t. the number of training samples used, ranging from 1 to 256.

IT: Instruction Tuning.

RT: Recommendation Tuning.

## Cross-domain Generalization

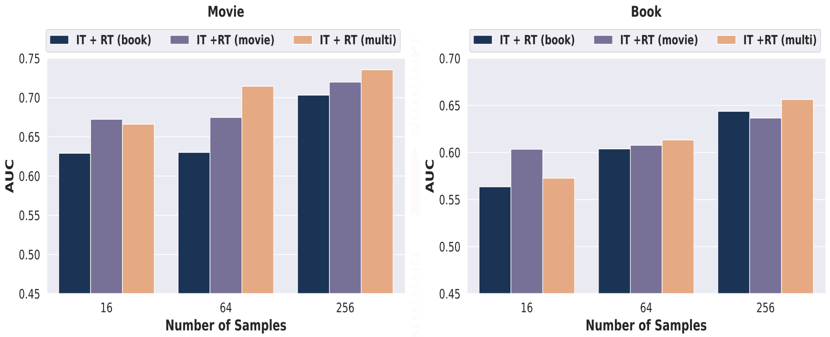


Fig. 2.12: Cross-domain performance of LLMs trained via TALLRec.

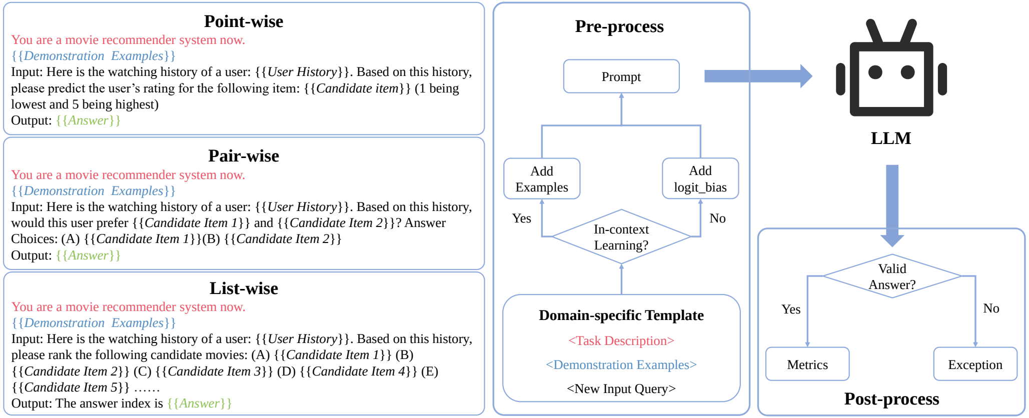
Models trained on Book perform worse than those trained on Movie when evaluated on the Book dataset?

## Contents

## ChatGPT for Recommendations

- Title: Uncovering ChatGPT's Capabilities in Recommender Systems [6]
- Author: Dai, Sunhao and Shao, Ninglu and Zhao, Haiyuan and Yu, Weijie and Si, Zihua and Xu, Chen and Sun, Zhongxiang and Zhang, Xiao and Xu, Jun (RUC)
- Published: RecSys 2023
- Use ICL to triger recommendation capabilities of LLMs.

## Evaluation Framework



**Point-wise**

You are a movie recommender system now.
{{*Demonstration Examples*}}
Input: Here is the watching history of a user: {{*User History*}}. Based on this history, please predict the user's rating for the following item: {{*Candidate item*}} (1 being lowest and 5 being highest)
Output: {{*Answer*}}

**Pair-wise**

You are a movie recommender system now.
{{*Demonstration Examples*}}
Input: Here is the watching history of a user: {{*User History*}}. Based on this history, would this user prefer {{*Candidate Item 1*}} and {{*Candidate Item 2*}}? Answer Choices: (A) {{*Candidate Item 1*}}(B) {{*Candidate Item 2*}}
Output: {{*Answer*}}

**List-wise**

You are a movie recommender system now.
{{*Demonstration Examples*}}
Input: Here is the watching history of a user: {{*User History*}}. Based on this history, please rank the following candidate movies: (A) {{*Candidate Item 1*}} (B) {{*Candidate Item 2*}} (C) {{*Candidate Item 3*}} (D) {{*Candidate Item 4*}} (E) {{*Candidate Item 5*}} ......
Output: The answer index is {{*Answer*}}

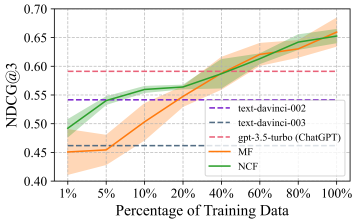
Fig. 2.13: The overall evaluation framework of LLMs for recommendation.

## Overall Performance

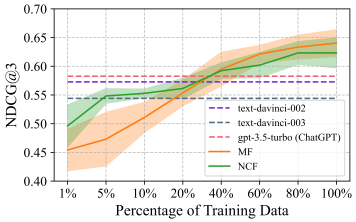
| Domain | Metric | random | pop | text-davinci-002 | | | text-davinci-003 | | | gpt-3.5-turbo (ChatGPT) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | point-wise | pair-wise | list-wise | point-wise | pair-wise | list-wise | point-wise | pair-wise | list-wise |
| Movie | Compliance Rate | - | - | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 99.98% | 100.00% |
| | NDCG@3 | 0.4262 | 0.4761 | 0.5416 | 0.5728 | 0.4990 | 0.4618 | 0.5441 | 0.5564 | **0.5912** | 0.5827 | 0.5785 |
| | MRR@3 | 0.3667 | 0.4103 | 0.4824 | 0.5071 | 0.4363 | 0.3998 | 0.4763 | 0.4950 | **0.5260** | 0.5162 | 0.5167 |
| Book | Compliance Rate | - | - | 99.96% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 99.98% | 99.80% |
| | NDCG@3 | 0.4262 | 0.4999 | 0.4889 | 0.5298 | 0.4290 | 0.4585 | 0.5293 | 0.4597 | 0.5075 | 0.5350 | **0.5395** |
| | MRR@3 | 0.3667 | 0.4340 | 0.4247 | 0.4646 | 0.3690 | 0.3993 | 0.4665 | 0.4040 | 0.4495 | 0.4774 | **0.4800** |
| Music | Compliance Rate | - | - | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 99.96% | 99.80% |
| | NDCG@3 | 0.4262 | 0.4094 | 0.4623 | 0.4681 | 0.4277 | 0.4732 | 0.5072 | 0.4506 | 0.5201 | 0.5439 | **0.5567** |
| | MRR@3 | 0.3667 | 0.3470 | 0.4030 | 0.4082 | 0.3750 | 0.4113 | 0.4448 | 0.4000 | 0.4605 | 0.4830 | **0.4950** |
| News | Compliance Rate | - | - | 99.80% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 100.00% | 99.60% |
| | NDCG@3 | 0.4262 | **0.5444** | 0.4483 | 0.4550 | 0.5059 | 0.4880 | 0.4892 | 0.4742 | 0.4826 | 0.4991 | 0.5094 |
| | MRR@3 | 0.3667 | **0.4840** | 0.3879 | 0.3936 | 0.4497 | 0.4271 | 0.4294 | 0.4173 | 0.4246 | 0.4354 | 0.4515 |

Fig. 2.14: Overall performance of different models on four datasets from different domains.

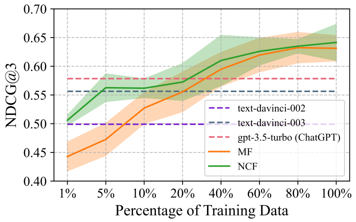
## Comparison with Collaborative Filtering Methods



(a) point-wise



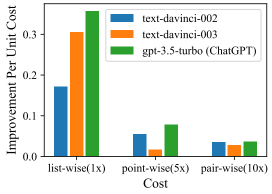
(b) pair-wise



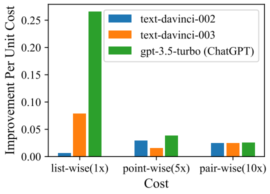
(c) list-wise

Fig. 2.15: Comparison with collaborative filtering models in terms of different percentages of training data on Movie dataset.

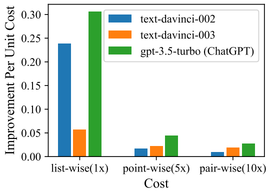
# Performance Scaling by Cost



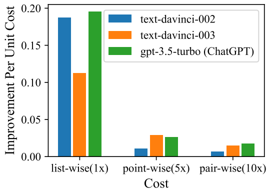
(a) Movie



(b) Book



(c) Music



(d) News

Fig. 2.16: Improvement of NDCG@3 per unit cost and five shots examples on four datasets.

## Exceptions

**Table 4: Case Study of Exceptions. The green is the answer of ChatGPT.**

| Case 1 | Case 2 |
|---|---|
| You are a movie recommender system now. {{*Examples*}} Input: Here is the watching history of a user: Aliens, E.T. the Extra-Terrestrial, Contact, The Matrix, X-Men. Based on this history, would this user prefer The Fox and the Hound or Steamboat Willie? Answer Choices: (A) The Fox and the Hound (B) Steamboat Willie Output: The answer index is N/A as neither option is relevant to the user's watching history. | You are a book recommender system now. {{*Examples*}} Input: Here is the reading history of a user: The Cellist of Sarajevo, After I'm Gone: A Novel, The Reason I Jump: The Inner Voice of a Thirteen-Year-Old Boy with Autism, The Serpent of Venice: A Novel, We Are All Completely Beside Ourselves: A Novel. Based on this history, would this user prefer The Secret Life of Bees or The Help? Answer Choices: (A) The Secret Life of Bees (B) The Help Output: The answer index is N/A. It is difficult to determine the user's preference based on this reading history as neither book is similar in genre or theme to the books they have read. |

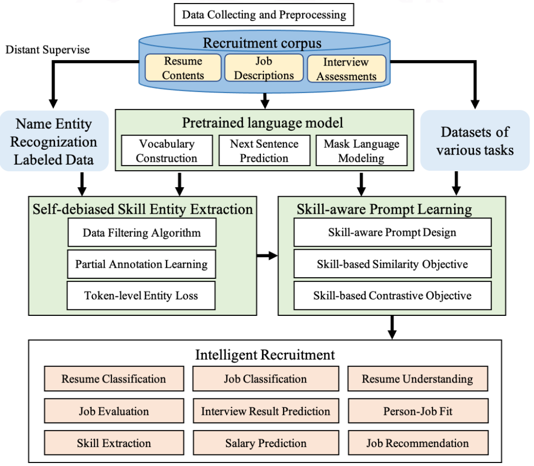
Fig. 2.17: Case Study of Exceptions. The green is the answer of ChatGPT.

## Contents

## RecruitPro

- Title: RecruitPro: A Pretrained Language Model with Skill-Aware Prompt Learning for Intelligent Recruitment [10]
- Author: Fang, Chuyu and Qin, Chuan and Zhang, Qi and Yao, Kaichun and Zhang, Jingshuai and Zhu, Hengshu and Zhuang, Fuzhen and Xiong, Hui (Beihang, Boss Zhipin, Baidu, etc.)
- Published: KDD2023
- They propose a unified representation model used for various downstream recruitment tasks.
  1. The recruitment corpus is different from general corpus in terms of text content and structure.
  2. A comprehensive benchmark dataset covering the primary tasks in the recruitment process is currently lacking.
  3. It is challenging to capture skill-related information in a unified representation model.

## Pre-training I



## Pre-training II

1. *Data Preprocessing*
   - This dataset encompasses more than 20 major industries in the recruitment field, and includes job descriptions, resume contents, and interview assessments.
   - Eliminate the recruitment data with significantly distinct text length (too long or too short) by box plot statistics and remove duplicate data.
   - After data filtering, the final datasets for job description, resume and interview evaluation contain *926,282, 785,244 and 187,983* records, respectively.
2. *Downstream Tasks*
   - *Job Classification*: to determine the corresponding job category given a piece of job description data.
   - *Resume Classification*: to predict the appropriate category for a given resume content.
   - *Resume Understanding*: to identify the category of each piece in resumes.
   - *Job Evaluation*: to determine the level of a job position (e.g., senior or junior).
   - *Interview Result Prediction*: to automatically evaluate interview assessments and make a judgment on the interview result.
   - *Person-Job Fit*
   - *Job Recommendation*
   - *Salary Prediction*

## Pre-training III

- *Skill Extraction*

3. *Model Pretraining*
   - Use the BPE (Byte Pair Encoding) algorithm for the English part of the text.
   - First segment the input text, and then the special tokens $[CLS]$ and $[SEP]$ will be added to the beginning and end of each text.

$$
\begin{aligned}
& x=[[CLS], \operatorname{seg}(\text{ text }),[SEP]], \\
& \operatorname{seg}(\text{ text })=x_1, x_2, \ldots, x_n, \\
& e_x=\operatorname{Embedding}(x), \\
& h_x=\operatorname{Transformers}\left(e_x\right), \\
& P(y \mid x)=F\left(W^T h_{[CLS]}+b\right),
\end{aligned}
$$

   - Objectives: mask language modeling and next sentence prediction.

## PLM Application to Tasks

- A well-designed prompt and a $[Mask]$ token will be as the additional input of the pretraining model.

$$\begin{aligned} x &= [[CLS]; \text{ prompt }; [MASK]; \text{seg( text )}; [SEP]], \\ e_x &= \text{ Embedding } (x), \\ h_x &= \text{ Transformers } (e_x), \\ P(y \mid x) &= F\left(W^T h_{[MASK]} + b\right), \end{aligned}$$

where $F$ is a mapping function related to the specific tasks.
- For the matching tasks, the job descriptions and resumes are split into shorter texts with fixed lengths and then obtain a series of text representations through our model.
- The final result is predicted by using a 2-layer LSTM network with a MLP layer.

## Skill-aware Prompt Learning I

*Skill-Aware Prompt Design.*

- Soft prompt learning and two types of prompts: skill-related and task-related prompt.
- The skill-related prompt focuses on extracting the semantic information of skill terms.
- The task-related prompt is scattered around the skill-related prompt and mask to mitigate the differences in the various tasks.

$$e_x = \left[e_{[CLS]}, e'_{p_{task_1}}, e'_{p_{skill}}, e'_{p_{task_2}}, e_{[MASK]}, e'_{p_{task_3}}, e_{text}, e_{[SEP]}\right]$$

$$\left[e'_{p_{task_1}}, e'_{p_{skill}}, e'_{p_{task_2}}, e'_{p_{task_3}}\right] = \mathrm{MLP}\left(\mathrm{BiLSTM}\left(e_{p_{task_1}}, e_{p_{skill}}, e_{p_{task_2}}, e_{p_{task_3}}\right)\right)$$

# Skill-aware Prompt Learning II

*Auxiliary Skill Sentence Construction.*

- We *extract* the skill terms from the input text sentences (*t-sen*) and concatenate all terms as new skill sentences (*s-sen*).

$$
\begin{aligned}
e_{x^{t\text{-}sen}} &= \left[e'^{t\text{-}sen}_{p_{task_1}}, e'^{t\text{-}sen}_{p_{skill}}, e'^{t\text{-}sen}_{p_{task_2}}, e^{t\text{-}sen}_{[MASK]}, e'^{t\text{-}sen}_{p_{task_3}}, e_{t\text{-}sen}\right] \\
h_{p_x^{t\text{-}sen}} &= \text{Transformer}\left(e'^{t\text{-}sen}_{p_{skill}}\right) \\
e_{x^{s\text{-}sen}} &= \left[e'^{s\text{-}sen}_{p_{task_1}}, e'^{s\text{-}sen}_{p_{skill}}, e'^{s\text{-}sen}_{p_{task_2}}, e^{s\text{-}sen}_{[MASK]}, e'^{s\text{-}sen}_{p_{task_3}}, e_{s\text{-}sen}\right] \\
h_{p_x^{s\text{-}sen}} &= \text{Transformer}\left(e'^{s\text{-}sen}_{p_{skill}}\right)
\end{aligned}
$$

## Skill-aware Prompt Learning III

*Multiview Loss Function*

- Sentence skill similarity loss

$$L_{sim} = -\sum_{k \in I} csim\left(h_{p_x^{t\text{-}sen_k}}, h_{p_x^{s\text{-}sen_k}}\right)$$

- Heterogeneous skill comparison loss to account for variations in skills across different instance classes and tasks

$$L_{com} = -\sum_{k \in I} \frac{1}{|P(k)|} \sum_{l \in P(k)} \log \frac{\exp\left(\text{csim}\left(h_{p_x^{s\text{-}sen_k}}, h_{p_x^{s\text{-}sen_l}}\right)/\tau\right)}{\sum_{o \in A(k)} \exp\left(c\,\text{sim}\left(h_{p_x^{s\text{-}sen_k}}, h_{p_x^{s\text{-}sen_o}}\right)\tau\right)}$$

where $A(k)$ indicate the set of instances in the same batch as the $k$-th instance, $P(k)$ represent the set of instances with the same label as the $k$-th instance in $A(k)$.

- Downstream task objective: MSE or Cross Entropy.

## Skill-aware Prompt Learning IV

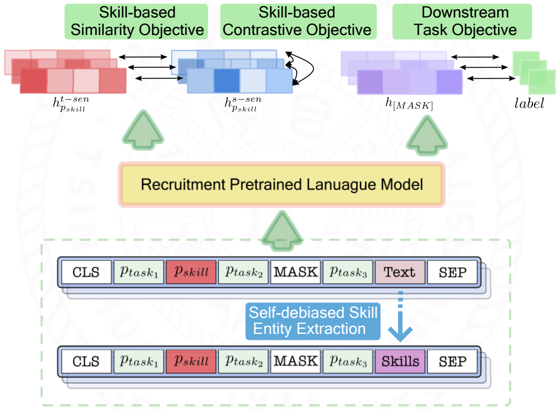


Fig. 3.1: The framework of skill-aware prompt learning.

## Self-debiased Skill Entity Extraction

1. *Data Filtering*. Distant supervised algorithm.
2. *Self-debiased Designing.*
   - Use the BIOES tagging schema to label the data and mark the uncertain token as $U$.
   - We incorporate a conditional random field (CRF) layer onto the pretrained language model for skill entity extraction.
   - We adopt the teacherstudent framework [17, 49] to further enhance the robustness of the model for noisy data.
   - CRF loss, entity and non-entity contrastive loss, and skill entity ratio loss

## Experiments

Baselines

1. BERT
2. RoBERTa
3. ERNIE
4. Ptuning on BERT.
5. RPLM
6. RecruitPro w/o ner, RecruitPro w/o sim, RecruitPro w/o com.

| Task | Category | #Class | #Num |
|---|---|---|---|
| Job third classification(JD3rdClass) | Classification | 591 | 926K |
| Job second classification(JD2ndClass) | | 101 | 926K |
| Job first classification(JD1stClass) | | 20 | 926K |
| Resume third classification(Re3rdClass) | | 87 | 671K |
| Resume second classification(Re2ndClass) | | 35 | 729K |
| Resume first classification(Re1stClass) | | 15 | 785K |
| Resume undertanding(Re-Un) | | 6 | 14K |
| Job evaluation(Job-Eva) | | 2 | 150K |
| Interview result prediction(IR-Pre) | | 2 | 188K |
| Person-job fit(P-J fit) | Matching | 2 | 3588K |
| Job recommendation(JD-Recom) | Ranking | - | 268K |
| Salary prediction(Salary-Pre) | Regression | - | 16K |
| Skill extraction | NER | 5 | 340K |

Fig. 3.2: The framework of skill-aware prompt learning.

**Table 2: Main results on 12 recruitment domain tasks.**

| | JD3rdClass | | JD2ndClass | | JD1stClass | | Re3rdClass | | Re2ndClass | | Re1stClass | | Re-Un | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ACC/% | F1/% | ACC/% | F1/% | ACC/% | F1/% | ACC/% | F1/% | ACC/% | F1/% | ACC/% | F1/% | ACC/% | F1/% |
| **BERT** | 67.64 | 66.31 | 75.96 | 75.08 | 82.90 | 82.36 | 29.79 | 28.76 | 34.51 | 33.98 | 45.57 | 44.57 | 79.50 | 79.51 |
| **Roberta** | 67.92 | 66.35 | 77.19 | 76.59 | 84.47 | 84.08 | 29.55 | 27.90 | 34.69 | 33.94 | 46.65 | 46.32 | 81.00 | 80.29 |
| **Ernie** | 68.02 | 66.65 | 76.96 | 76.03 | 84.73 | 84.22 | 29.52 | 27.43 | 36.04 | 34.91 | 46.45 | 45.60 | 81.00 | 80.74 |
| **Ptuning** | 67.66 | 66.66 | 76.17 | 76.56 | 85.47 | 84.59 | 29.56 | 28.24 | 36.02 | 34.94 | 45.28 | 44.53 | 80.67 | 80.49 |
| **RPLM** | 68.57 | 67.40 | 77.33 | 76.63 | 85.20 | 84.61 | 29.59 | 28.49 | 36.67 | 35.66 | 46.48 | 45.36 | 79.83 | 79.65 |
| **RecruitPro** | **68.94** | **67.63** | 77.62 | 76.81 | **85.63** | **84.89** | **30.31** | **28.92** | 37.57 | 36.85 | 47.69 | **47.20** | **82.50** | **82.18** |
| - w/o ner | 68.89 | **67.63** | **77.81** | **77.25** | 85.19 | 84.44 | 29.37 | 28.05 | 37.38 | 36.51 | 47.48 | 46.64 | 81.50 | 81.47 |
| - w/o sim | 68.59 | 67.12 | 77.26 | 76.57 | 85.40 | 84.40 | 29.61 | 27.97 | 37.58 | 36.84 | 47.53 | 46.59 | 82.17 | 81.85 |
| - w/o com | 68.51 | 67.24 | 77.29 | 76.64 | 85.53 | 84.62 | 29.30 | 27.52 | **37.67** | **37.21** | **47.75** | 46.85 | 81.17 | 80.94 |

| | Job-Eva | | | IR-Pre | | | P-J fit | | | Salary-Pre | | JD-Recom | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ACC/% | F1/% | AUC/% | ACC/% | F1/% | AUC/% | ACC/% | F1/% | AUC/% | RMSE | MAE | MRR | Hits@1 | Hits@10 |
| **BERT** | 70.34 | 70.34 | 77.02 | 89.94 | 89.94 | 96.49 | 71.31 | 71.23 | 75.47 | 1.0130 | 0.6900 | 0.3328 | 0.1906 | 0.6412 |
| **Roberta** | 70.97 | 70.93 | 78.66 | 90.36 | 90.36 | 96.69 | 69.56 | 69.55 | 76.92 | 0.9745 | 0.7534 | 0.3805 | 0.2281 | 0.6925 |
| **Ernie** | 71.16 | 71.11 | 77.76 | 90.17 | 90.17 | 96.70 | 68.48 | 68.13 | 75.79 | 0.9981 | 0.6782 | 0.3607 | 0.2053 | 0.6813 |
| **Ptuning** | 70.97 | 70.91 | 77.90 | 89.90 | 89.79 | 96.45 | 71.11 | 70.96 | 79.33 | 0.9840 | **0.6420** | 0.3242 | 0.1729 | 0.6447 |
| **RPLM** | 71.47 | 71.47 | 79.04 | 90.17 | 90.17 | 96.79 | 71.35 | 71.24 | 75.47 | 0.9986 | 0.7436 | 0.4031 | 0.2525 | 0.7012 |
| **RecruitPro** | 71.64 | 71.58 | **79.22** | **90.83** | **90.83** | 96.95 | **72.05** | **72.05** | **80.25** | 0.9818 | 0.7084 | **0.4328** | **0.2842** | **0.7306** |
| - w/o ner | 71.12 | 71.06 | 77.97 | 90.52 | 90.52 | 96.79 | 71.80 | 71.79 | 79.91 | 0.9814 | 0.7217 | 0.4310 | 0.2800 | 0.7291 |
| - w/o sim | **71.92** | **71.88** | 79.11 | 90.68 | 90.68 | **96.97** | 71.81 | 71.76 | 80.08 | **0.9679** | 0.6949 | 0.4279 | 0.2765 | 0.7279 |
| - w/o com | 71.59 | 71.38 | 79.07 | 90.58 | 90.58 | 96.95 | 71.24 | 71.22 | 79.66 | 1.0367 | 0.7244 | 0.4247 | 0.2735 | 0.7254 |

## Contents

## Generative Job Recommendations I

- Title: Generative Job Recommendations with Large Language Model [52]
- Author: Zheng, Zhi and Qiu, Zhaopeng and Hu, Xiao and Wu, Likang and Zhu, Hengshu and Xiong, Hui (USTC, HKUST)
- Published: ArXiv
- They propose a novel user-centered GeneratIve job Recommendation paradigm based on LLM (GIRL).
- Limitations of current models
  1. Poor explainability.
  2. Discriminative models cannot be comprehensive career AI advisors.
  3. The existence of the considerable semantic gap between CVs and JDs has resulted in the underwhelming performance of traditional methods.

## Generative Job Recommendations II

**Definition 3.1 (Generative Job Recommendation)**

*Given a job seeker $s$ with the corresponding $C$, the goal of generative job recommendation is to train a generator $\mathcal{G}$ , which can generate a suitable JD for this user, i.e., $\mathcal{G} : C \to J'$.*

**Definition 3.2 (Generation-Enhanced Job Recommendation)**

*Given a job seeker $s$ with the corresponding $C$, a job $j$ with the corresponding $J$, and the generated $J'$ , the goal of generation-enhanced job recommendation is to train a model $\mathcal{M}$, which can calculate the matching score between $s$ and $j$, i.e., $\mathcal{M} : C, J, J' \to \mathbb{R}$.*

## Generative Job Recommendations III

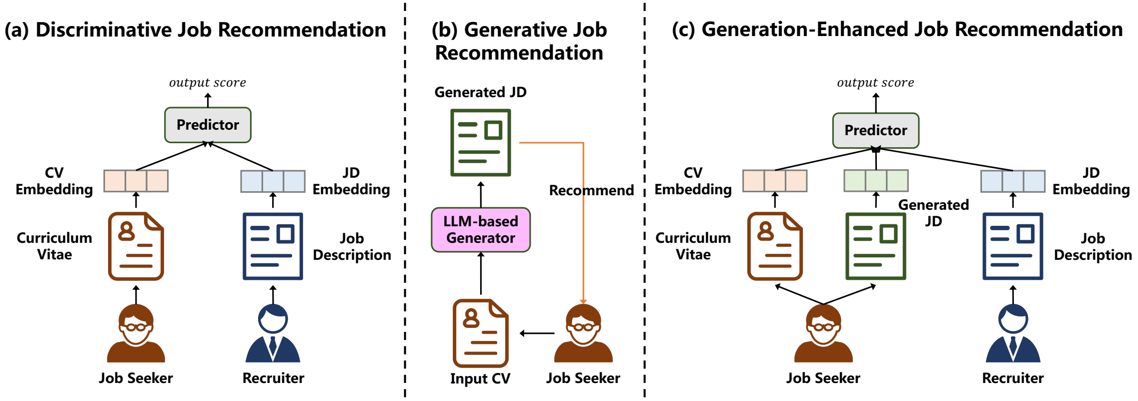
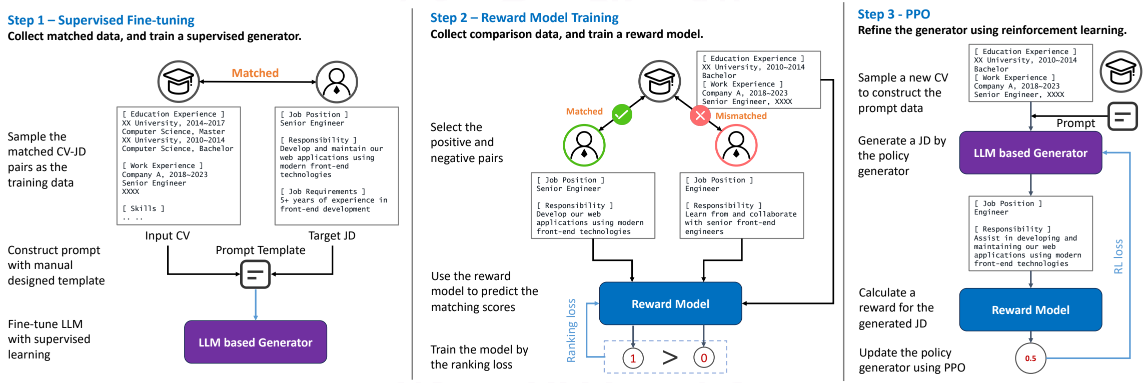


Fig. 3.3: Schematic diagram of three distinct job recommendation paradigms.

# Generative Recommendation Framework I



Step 1 – Supervised Fine-tuning
Collect matched data, and train a supervised generator.
Matched
Sample the matched CV-JD pairs as the training data
[ Education Experience ]
XX University, 2014~2017
Computer Science, Master
XX University, 2010~2014
Computer Science, Bachelor
[ Work Experience ]
Company A, 2018~2023
Senior Engineer
XXXX
[ Skills ]
.. ..
[ Job Position ]
Senior Engineer
[ Responsibility ]
Develop and maintain our web applications using modern front-end technologies
[ Job Requirements ]
5+ years of experience in front-end development
Input CV
Target JD
Construct prompt with manual designed template
Prompt Template
Fine-tune LLM with supervised learning
LLM based Generator
Step 2 – Reward Model Training
Collect comparison data, and train a reward model.
Select the positive and negative pairs
[ Education Experience ]
XX University, 2010~2014
Bachelor
[ Work Experience ]
Company A, 2018~2023
Senior Engineer, XXXX
Matched
Mismatched
[ Job Position ]
Senior Engineer
[ Responsibility ]
Develop our web applications using modern front-end technologies
[ Job Position ]
Engineer
[ Responsibility ]
Learn from and collaborate with senior front-end engineers
Use the reward model to predict the matching scores
Reward Model
Ranking loss
Train the model by the ranking loss
1
>
0
Step 3 - PPO
Refine the generator using reinforcement learning.
Sample a new CV to construct the prompt data
[ Education Experience ]
XX University, 2010~2014
Bachelor
[ Work Experience ]
Company A, 2018~2023
Senior Engineer, XXXX
Prompt
Generate a JD by the policy generator
LLM based Generator
[ Job Position ]
Engineer
[ Responsibility ]
Assist in developing and maintaining our web applications using modern front-end technologies
RL loss
Calculate a reward for the generated JD
Reward Model
Update the policy generator using PPO
0.5

Fig. 3.4: The training framework of the generative recommendation model.

## Generative Recommendation Framework II

1. *Supervised Fine-tuning*
   - Given a specific job seeker $s$ with the CV $C$ and a job $j$ with the JD $J$, we first build a prompt $T$ to describe the generation task.
   - Then, we propose to train the generator with the casual language model pre-training task.

Human:
Please generate a suitable job description based on the curriculum vitae of the following candidate:
Basic information: The candidate is a 27-year-old female with a bachelor's degree, majoring in Software Engineering.
personal profile:
1. She has research experience in the field of data processing and computation, and development experience with components from ElasticSearch, Hbase, and the Flink ecosystem.
2. She possesses strong abilities to read and understand source code, having reviewed core source code from JDK8, Spring, Spring Boot, and Netty.

Assistant:
Job title: Java Developer
Salary range: 25k-40k* 15
Responsibilities:
1. Support the recommendation product and algorithm departments, rapidly iterate daily needs, complete effect analysis and continuous improvements.
2. Track industry developments, timely introduction of external advanced experiences.

Qualifications:
1. Proficient in Java programming basics, with excellent abilities and experience in Java IO and multithreading programming.
2. In-depth understanding of JVM, JVM tuning experience, and experience with distributed systems and concurrency scenarios are preferred.
3. Proficient in applying mainstream development frameworks and open-source systems such as Spring Boot, MyBatis, MySQL, Redis, ES, Kafka, etc.
4. Good stress resistance, communication, learning, collaboration skills, and a strong sense of responsibility.
5. Prior experience in recommendation/search engineering development in Internet companies is preferred.

## Generative Recommendation Framework III

2. *Reward Model Training*
    - A reward model $\mathcal{U}$ that can predict the matching score between a CV-JD pair, i.e., $\mathcal{U} : (C, J) \rightarrow \mathbb{R}$.
    - The architecture of $\mathcal{U}$ is similar to that of the generator model $\mathcal{G}$ , but it has a linear prediction head that outputs scalar values.
3. *Reinforcement Learning*: Proximal Policy Optimization.

## Generation-enhanced Recommendation Framework

1. *Basic Recommendation Model*
   - Given a job seeker $s$ with the corresponding CV $C$, and a job $j$ with the corresponding JD $J$, we first need to get the text embedding based on a text encoder as:

     $$\mathbf{c} = Encoder(C), \quad \mathbf{j} = Encoder(J)$$

   - Then we get the matching score by MLP predictor $score = MLP([\mathbf{c}; \mathbf{j}])$ or dot predictor $score = \mathbf{c} \cdot \mathbf{j}$
2. *Enhanced Recommendation Model*
   - Get the text embedding of the generated JD $J'$

     $$\mathbf{j}' = Encoder(J')$$

   - MLP predictor: $score = MLP([\mathbf{c}; \mathbf{j}; \mathbf{j}'])$
   - Dot predictor:

     $$\mathbf{c}' = MLP([\mathbf{c}; \mathbf{j}'])$$
     $$score = \mathbf{c}' \cdot \mathbf{j}$$

## Experiment Settings I

| Description | Number |
|---|---|
| # of data for supervised fine-tuning | 153,006 |
| # of data for reward model training | 303,929 |
| # of data for reinforcement learning | 37,600 |
| # of data in training set for enhanced recommendation | 37,158 |
| # of data in validation set for enhanced recommendation | 4,542 |
| # of data in test set for enhanced recommendation | 6,300 |

Fig. 3.5: Statistics of the Datasets.

1. Generation Quality (evaluated by ChatGPT).
   - GIRL, GIRL-SFT and Other LLMs (BELLE-7b, BLOOMZ-7b, and LLAMA-7b)
   - ChatGPT outputs ”Win”, ”Tie”, and ”Lose”
2. Performance Comparison
   - Base: choose BERT as the text encoder.
   - GIRL-SFT, GIRL

## Results I

| Predictor | Model | AUC(↑) | LogLoss(↓) |
|---|---|---|---|
| MLP | Base | 0.6349 | 0.4043 |
| | GIRL-SFT | 0.6438(+1.4%) | 0.3973(+1.7%) |
| | GIRL | **0.6476**(+2.0%) | **0.3908**(+3.3%) |
| Dot | Base | 0.6258 | 0.4964 |
| | GIRL-SFT | 0.6291(+0.5%) | 0.3688(+20.3%) |
| | GIRL | **0.6436**(+2.8%) | **0.3567**(+28.1%) |

| Predictor | Model | AUC(↑) | LogLoss(↓) |
|---|---|---|---|
| MLP | Base | 0.6198 | 0.4270 |
| | GIRL-SFT | 0.6293(+1.5%) | **0.4154**(+2.8%) |
| | GIRL | **0.6347**(+2.4%) | 0.4229(+1.0%) |
| Dot | Base | 0.6136 | 0.5233 |
| | GIRL-SFT | 0.6231(+1.5%) | 0.3827(+26.9%) |
| | GIRL | **0.6457**(+5.2%) | **0.3673**(+29.8%) |

Fig. 3.6: Overall performance of different models (left) and results under cold-start conditions (right).

## Results II

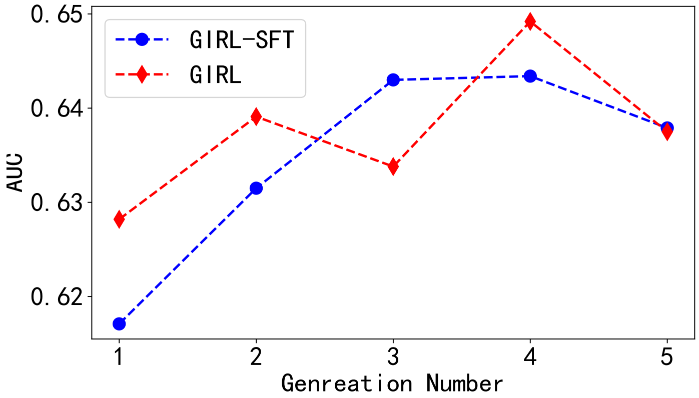


Fig. 3.7: Performance of different models with different generation number.

## Summary I

1. *Model Utilization.* How to design better prompts? Multi-turn Prompting?
2. *Safety & Alignment.* RLHF heavily relies on high-quality human feedback data from professional labelers.
3. *LLM for RecSys.* It has attracted a lot of research, but there is still no obvious conclusion on how to effectively use LLM in RecSys.
4. *Controllability.*
5. *LLM for Explanations.*
6. *LLM for Agents.*

## Reference I

[1] Bao, K., Zhang, J., Zhang, Y., Wang, W., Feng, F., and He, X.
Tallrec: An effective and efficient tuning framework to align large language model with recommendation.
*arXiv preprint arXiv:2305.00447* (2023).

[2] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.
Language models are few-shot learners.
*Advances in neural information processing systems 33* (2020), 1877–1901.

[3] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al.
Palm: Scaling language modeling with pathways.
*arXiv preprint arXiv:2204.02311* (2022).

[4] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al.
Scaling instruction-finetuned language models.
*arXiv preprint arXiv:2210.11416* (2022).

[5] Cui, Z., Ma, J., Zhou, C., Zhou, J., and Yang, H.
M6-rec: Generative pretrained language models are open-ended recommender systems.
*arXiv preprint arXiv:2205.08084* (2022).

[6] Dai, S., Shao, N., Zhao, H., Yu, W., Si, Z., Xu, C., Sun, Z., Zhang, X., and Xu, J.
Uncovering chatgpt's capabilities in recommender systems.
*arXiv preprint arXiv:2305.02182* (2023).

[7] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L.
Qlora: Efficient finetuning of quantized llms.
*arXiv preprint arXiv:2305.14314* (2023).

[8] Fan, W., Zhao, Z., Li, J., Liu, Y., Mei, X., Wang, Y., Tang, J., and Li, Q.
Recommender systems in the era of large language models (llms).
*arXiv preprint arXiv:2307.02046* (2023).

## Reference II

[9] Fan, Z., Liu, Z., Heinecke, S., Zhang, J., Wang, H., Xiong, C., and Yu, P. S.
Zero-shot item-based recommendation via multi-task product knowledge graph pre-training.
*arXiv preprint arXiv:2305.07633* (2023).

[10] Fang, C., Qin, C., Zhang, Q., Yao, K., Zhang, J., Zhu, H., Zhuang, F., and Xiong, H.
Recruitpro: A pretrained language model with skill-aware prompt learning for intelligent recruitment.
In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2023), pp. 3991–4002.

[11] Friedman, L., Ahuja, S., Allen, D., Tan, T., Sidahmed, H., Long, C., Xie, J., Schubiner, G., Patel, A., Lara, H., et al.
Leveraging large language models in conversational recommender systems.
*arXiv preprint arXiv:2305.07961* (2023).

[12] Fu, Y., Peng, H., and Khot, T.
How does gpt obtain its ability? tracing emergent abilities of language models to their sources.
*Yao Fu's Notion* (2022).

[13] Gao, Y., Sheng, T., Xiang, Y., Xiong, Y., Wang, H., and Zhang, J.
Chat-rec: Towards interactive and explainable llms-augmented recommender system.
*arXiv preprint arXiv:2303.14524* (2023).

[14] Geng, S., Liu, S., Fu, Z., Ge, Y., and Zhang, Y.
Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5).
In *Proceedings of the 16th ACM Conference on Recommender Systems* (2022), pp. 299–315.

[15] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K.
A survey of quantization methods for efficient neural network inference.
In *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[16] Hou, Y., He, Z., McAuley, J., and Zhao, W. X.
Learning vector-quantized item representation for transferable sequential recommenders.
In *Proceedings of the ACM Web Conference 2023* (2023), pp. 1162–1171.

## Reference III

[17] HOULSBY, N., GIURGIU, A., JASTRZEBSKI, S., MORRONE, B., DE LAROUSSILHE, Q., GESMUNDO, A., ATTARIYAN, M., AND GELLY, S.
Parameter-efficient transfer learning for nlp.
In *International Conference on Machine Learning* (2019), PMLR, pp. 2790–2799.

[18] HU, E. J., WALLIS, P., ALLEN-ZHU, Z., LI, Y., WANG, S., WANG, L., CHEN, W., ET AL.
Lora: Low-rank adaptation of large language models.
In *International Conference on Learning Representations* (2021).

[19] HUA, W., GE, Y., XU, S., JI, J., AND ZHANG, Y.
Up5: Unbiased foundation model for fairness-aware recommendation.
*arXiv preprint arXiv:2305.12090* (2023).

[20] HUA, W., XU, S., GE, Y., AND ZHANG, Y.
How to index item ids for recommendation foundation models.
*arXiv preprint arXiv:2305.06569* (2023).

[21] KOJIMA, T., GU, S. S., REID, M., MATSUO, Y., AND IWASAWA, Y.
Large language models are zero-shot reasoners.
*Advances in neural information processing systems 35* (2022), 22199–22213.

[22] LESTER, B., AL-RFOU, R., AND CONSTANT, N.
The power of scale for parameter-efficient prompt tuning.
In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 3045–3059.

[23] LI, L., ZHANG, Y., LIU, D., AND CHEN, L.
Large language models for generative recommendation: A survey and visionary discussions.
*arXiv preprint arXiv:2309.01157* (2023).

# Reference IV

[24] Li, X. L., and Liang, P.
Prefix-tuning: Optimizing continuous prompts for generation.
In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2021), pp. 4582–4597.

[25] Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., and Stone, P.
Llm+ p: Empowering large language models with optimal planning proficiency.
*arXiv preprint arXiv:2304.11477* (2023).

[26] Liu, J., Liu, C., Lv, R., Zhou, K., and Zhang, Y.
Is chatgpt a good recommender? a preliminary study.
*arXiv preprint arXiv:2304.10149* (2023).

[27] Liu, J., Shen, D., Zhang, Y., Dolan, W. B., Carin, L., and Chen, W.
What makes good in-context examples for gpt-3?
In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures* (2022), pp. 100–114.

[28] Nguyen, H.-T.
A brief report on lawgpt 1.0: A virtual legal assistant based on gpt-3.
*arXiv preprint arXiv:2302.05729* (2023).

[29] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.
Training language models to follow instructions with human feedback, 2022.
*URL https://arxiv. org/abs/2203.02155 13* (2022).

[30] Petrov, A. V., and Macdonald, C.
Generative sequential recommendation with gptrec.
*arXiv preprint arXiv:2306.11114* (2023).

# Reference V

[31] Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y.
Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters.
In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 3505–3506.

[32] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T.
Toolformer: Language models can teach themselves to use tools.
*arXiv preprint arXiv:2302.04761* (2023).

[33] Shen, T., Li, J., Bouadjenek, M. R., Mai, Z., and Sanner, S.
Towards understanding and mitigating unintended biases in language model-driven conversational recommendation.
*Information Processing & Management 60*, 1 (2023), 103139.

[34] Shin, K., Kwak, H., Kim, K.-M., Kim, M., Park, Y.-J., Jeong, J., and Jung, S.
One4all user representation for recommender systems in e-commerce.
*arXiv preprint arXiv:2106.00573* (2021).

[35] Shinn, N., Cassano, F., Labash, B., Gopinath, A., Narasimhan, K., and Yao, S.
Reflexion: Language agents with verbal reinforcement learning.
*arXiv preprint arXiv:2303.11366* (2023).

[36] Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B.
Megatron-lm: Training multi-billion parameter language models using model parallelism.
*arXiv preprint arXiv:1909.08053* (2019).

[37] Singhal, K., Azizi, S., Tu, T., Mahdavi, S. S., Wei, J., Chung, H. W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al.
Large language models encode clinical knowledge.
*Nature* (2023), 1–9.

[38] Wang, L., Zhang, J., Chen, X., Lin, Y., Song, R., Zhao, W. X., and Wen, J.-R.
Recagent: A novel simulation paradigm for recommender systems.
*arXiv preprint arXiv:2306.02552* (2023).

## Reference VI

[39] Wang, X., Tang, X., Zhao, W. X., Wang, J., and Wen, J.-R.
Rethinking the evaluation for conversational recommendation in the era of large language models.
*arXiv preprint arXiv:2305.13112* (2023).

[40] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al.
Emergent abilities of large language models.
*arXiv preprint arXiv:2206.07682* (2022).

[41] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al.
Chain-of-thought prompting elicits reasoning in large language models.
*Advances in Neural Information Processing Systems 35* (2022), 24824–24837.

[42] Wu, C., Wu, F., Qi, T., Lian, J., Huang, Y., and Xie, X.
Ptum: Pre-training user model from unlabeled user behaviors via self-supervision.
*arXiv preprint arXiv:2010.01494* (2020).

[43] Wu, L., Qiu, Z., Zheng, Z., Zhu, H., and Chen, E.
Exploring large language model for graph data understanding in online job recommendations.
*arXiv preprint arXiv:2307.05722* (2023).

[44] Wu, Y., Xie, R., Zhu, Y., Zhuang, F., Zhang, X., Lin, L., and He, Q.
Personalized prompts for sequential recommendation.
*arXiv preprint arXiv:2205.09666* (2022).

[45] Xiong, H., Wang, S., Zhu, Y., Zhao, Z., Liu, Y., Wang, Q., and Shen, D.
Doctorglm: Fine-tuning your chinese doctor is not a herculean task.
*arXiv preprint arXiv:2304.01097* (2023).

[46] Yang, H., Liu, X.-Y., and Wang, C. D.
Fingpt: Open-source financial large language models.
*arXiv preprint arXiv:2306.06031* (2023).

## Reference VII

[47] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K.
Tree of thoughts: Deliberate problem solving with large language models.
*arXiv preprint arXiv:2305.10601* (2023).

[48] Zhang, H., Chen, J., Jiang, F., Yu, F., Chen, Z., Li, J., Chen, G., Wu, X., Zhang, Z., Xiao, Q., et al.
Huatuogpt, towards taming language model to be a doctor.
*arXiv preprint arXiv:2305.15075* (2023).

[49] Zhang, J., Bao, K., Zhang, Y., Wang, W., Feng, F., and He, X.
Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation.
*arXiv preprint arXiv:2305.07609* (2023).

[50] Zhang, J., Xie, R., Hou, Y., Zhao, W. X., Lin, L., and Wen, J.-R.
Recommendation as instruction following: A large language model empowered recommendation approach.
*arXiv preprint arXiv:2305.07001* (2023).

[51] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al.
A survey of large language models.
*arXiv preprint arXiv:2303.18223* (2023).

[52] Zheng, Z., Qiu, Z., Hu, X., Wu, L., Zhu, H., and Xiong, H.
Generative job recommendations with large language model.
*arXiv preprint arXiv:2307.02157* (2023).