# 提示词工程考试说明（Prompt Engineering）

## 任务背景

在大数据时代，推荐系统成为各大互联网平台不可或缺的核心技术。传统推荐系统通常包括以下环节：

\* \*\*数据收集\*\*

\* \*\*召回（Recall）\*\*：从海量候选集中快速筛选潜在相关物品。

\* \*\*排序（Ranking）\*\*：对召回结果进行精细排序。

\* \*\*重排（Re-ranking）\*\*：对排序后的候选集进一步精细化处理，直接影响推荐效果。

\* \*\*展示（Display）\*\*

序列推荐（Sequential Recommendation）考虑用户历史行为的时序信息，预测用户下一个行为，捕捉用户兴趣的动态变化，提高推荐的精准性和时效性。

大语言模型（LLM，如GPT、Claude）在推荐系统中展现出以下独特优势：

\* 强大的文本理解和语义分析能力

\* 跨模态理解能力

\* 零样本/少样本学习能力

\* 较强的可解释性

\* 良好的通用性

提示词工程（Prompt Engineering）通过精心设计输入提示（prompts），引导LLM生成符合预期输出的内容，涉及：

\* 提示结构设计

\* 上下文信息组织

\* 任务分解

\* 指令精确表达等

有效的Prompt能帮助LLM更准确理解用户偏好，并生成更个性化的推荐结果。

---

## 考试内容与要求

本次考核聚焦于利用\*\*提示词工程（Prompt Engineering）\*\*引导LLM完成推荐系统的\*\*重排任务（Re-ranking）\*\*：

\* 考生将获得验证集数据文件：`val.jsonl`

\* 每条数据代表用户历史电影观看记录，预测用户下一步可能观看的电影。

\* 要求设计Prompt引导LLM对候选电影（candidates）进行精准排序。

### 验证集数据字段说明（`val.jsonl`）：

```json

{

"user\_id": 5737,

"item\_list": [

[1836, "Last Days of Disco, The"],

[3565, "Where the Heart Is"],

...

],

"target\_item": [1893, "Beyond Silence"],

"candidates": [

[2492, "20 Dates"],

[684, "Windows"],

[1893, "Beyond Silence"],

...

]

}

```

\* `item\_list`：用户历史观看的电影列表（按时间顺序排列）,越靠后表示越近期观看。

\* `target\_item`：用户实际下一部观看的电影（ground truth）。

\* `candidates`：召回阶段提供的电影候选集（包含用户实际观看的下一部电影），一般约20个电影，需重排排序。

---

## 评价指标：NDCG\@K (Normalized Discounted Cumulative Gain)

衡量模型推荐性能的指标，范围 \[0, 1]，越接近1表示推荐质量越高。

### 计算方法：

\* 预测排序列表为 $p = [p\_1, p\_2, ..., p\_k]$

\* 实际观看电影为 $g$

1. \*\*相关性评分\*\*：

\* 若 $p\_i = g$，则 $rel\_i = 1$；否则，$rel\_i = 0$。

2. \*\*折损累积收益 (DCG)\*\*：

$$

DCG@k = \sum\_{i=1}^{k} \frac{rel\_i}{\log\_2(i+1)}

$$

3. \*\*理想折损累积收益 (IDCG)\*\*：

理想情况下，只有一个相关项（置于首位）：

$$

IDCG@k = \frac{1}{\log\_2(1+1)} = 1

$$

4. \*\*归一化折损累积收益 (NDCG)\*\*：

$$

NDCG@k = \frac{DCG@k}{IDCG@k} = DCG@k

$$

### 代码示例：

```python

import math

def calculate\_ndcg\_for\_sample(predicted\_list, ground\_truth\_item, k=10):

predicted\_list = predicted\_list[:k]

relevance = [1 if item\_id == ground\_truth\_item else 0 for item\_id in predicted\_list]

dcg = sum(rel / math.log2(i + 2) for i, rel in enumerate(relevance))

idcg = 1.0 # 只有一个相关项置于首位的情况

ndcg = dcg / idcg if idcg > 0 else 0

return ndcg

# 示例使用

predicted\_list = [111, 1893, 684, 2492, 3654, 2422, 176, 1629, 229, 3155]

ground\_truth\_item = 1893

ndcg = calculate\_ndcg\_for\_sample(predicted\_list, ground\_truth\_item, k=10)

print(f"NDCG@10 = {ndcg}") # 输出: NDCG@10 = 0.63093

```

---

## 考核方式

\* 使用 \*\*OpenAI API\*\* (Chat API模式) 构造 Prompt。

\* 完成以下两个函数（Python标准库）：

### 函数一：构造提示词 (`construct\_prompt`)

```python

def construct\_prompt(d):

"""

构造用于大语言模型的提示词

参数:

d (dict): jsonl数据文件的一行，解析成字典后的变量

返回:

list: OpenAI API的message格式列表

示例: [{"role": "system", "content": "系统提示内容"},

{"role": "user", "content": "用户提示内容"}]

"""

# 实现提示词构造逻辑

```

### 函数二：解析模型输出 (`parse\_output`)

```python

def parse\_output(text):

"""

解析大语言模型的输出文本，提取推荐重排列表

参数:

text (str): 大语言模型在设计prompt下的输出文本

返回:

list: 从输出文本解析出的电影ID列表（python列表格式）

示例: [1893, 3148, 111, ...]

"""

# 实现输出解析逻辑

```

## 评分标准（共两部分）

\* \*\*推荐性能客观得分（70%）\*\*：

\* 在私有测试集 (`test.jsonl`) 上的 NDCG\@10 排名与赋分

\* 测试统一使用 \*\*DeepSeek-V3 模型\*\*，temperature设为0

\* \*\*提示词主观评价得分（30%）\*\*：

\* 专家评分，内容包括：

\* 提示词创新性

\* 合理性

\* 可解释性