实验报告

李丁 PB23111595　罗浩民 PB23111599

2025 年 12 月 15 日

目录

1 问题定义

本赛题是一个新闻推荐任务，目标是预测用户未来点击的新闻文章。数据集包含 30 万用户、近 300 万次点击记录和 36 万多篇不同的新闻文章。任务要求为每个用户预测 Top-5 最可能点击的文章，评估指标为 MRR（Mean Reciprocal Rank）。

数据特点：

- 训练集：20 万用户的点击日志
- 测试集 A：5 万用户的点击日志
- 测试集 B：5 万用户的点击日志
- 数据包含用户点击行为、文章信息、文章 embedding 向量等

挑战：

1. 数据稀疏性：用户-文章交互矩阵非常稀疏
2. 类别不平衡：正负样本比例约为 1:196
3. 冷启动问题：新用户和新文章的推荐
4. 时效性：需要考虑时间因素对用户兴趣的影响

2 做题思路

我们采用经典的" 召回 + 排序" 两阶段推荐系统架构，通过多次迭代优化逐步提升模型性能。

2.1 整体流程

我们采用经典的" 召回 + 排序" 两阶段推荐系统架构，整体流程如下：

数据预处理 → 召回阶段 → 特征工程 → 排序阶段 → 结果生成

- **召回阶段**：从海量文章中快速筛选出可能相关的候选文章（约 154 篇/用户）
- **排序阶段**：对候选文章进行精细排序，预测用户最可能点击的 Top-5 文章

2.2 迭代优化过程

我们通过四次迭代逐步优化模型性能，每次迭代都有明确的改进目标：

表 1: 迭代优化过程总结

版本	召回策略	排序策略与特征	MRR@5
第一版	仅 ItemCF 协同过滤	LightGBM + 基础特征	≈ 0.28
第二版	ItemCF + Binetwork + W2V 融合	LightGBM + 相似度特征 + 时间特征	≈ 0.30
第三版	优化融合策略（Min-Max 标准化）	LightGBM + 用户行为特征 + 时间序列特征	≈ 0.31
第四版	保持多路召回融合	LightGBM + LambdaMART + 列表级别特征	≈ 0.31

关键改进点：

- **召回阶段**：从单一方法到多路召回融合，提升召回覆盖率
- **特征工程**：从基础特征到 30 个精心设计的特征，包括相似度、时间、用户行为等
- **排序方法**：从 Pointwise 到 Listwise，直接优化排序指标
- **工程优化**：向量化计算、交叉验证、样本权重等，提升训练效率和模型性能

3 数据分析

3.1 数据概览

数据集规模庞大，包含 30 万用户、近 300 万次点击记录和 36 万多篇不同的新闻文章。具体数据分布如下：

表 2: 数据概览

数据项	数量/比例
训练集点击记录	1,112,623 条
测试集点击记录	518,010 条
验证集用户数	50,000
召回后样本数	13,900,088（平均每个用户召回约 154 篇文章）
正负样本比例	1:196（严重不平衡）

3.2 特征分析

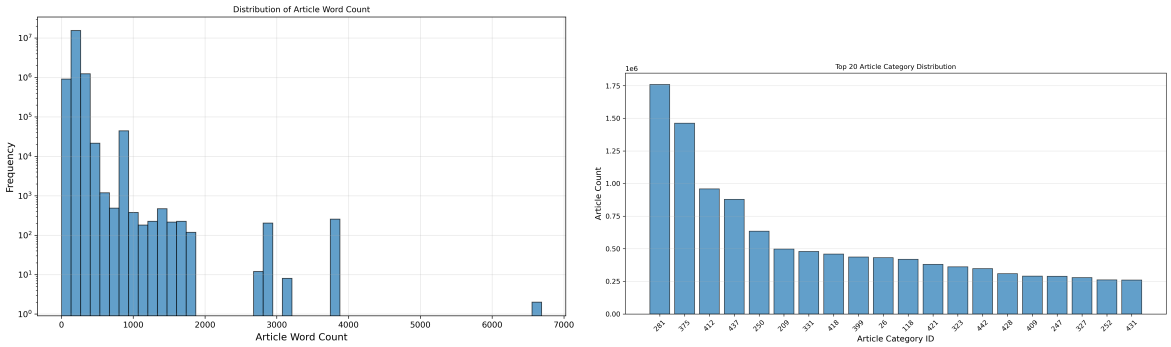
我们构建了 30 个特征，主要分为文章基础特征、召回相似度特征、用户行为特征和列表级别特征四大类。下面用表格形式总结主要特征，便于理解：

3.2.1 文章基础特征

文章基础特征直接反映文章本身的属性，这些特征简单但有效，能帮助模型理解文章的基本信息。

表 3: 文章基础特征总结

特征名称	统计值	业务含义
文章字数（words_count）	均值 200.83，中位数 196	反映用户阅读习惯，不同用户对文章长度偏好不同
文章类别（category_id）	-	反映用户对不同类别文章的偏好，不同类别受欢迎程度差异大
文章创建时间（created_at_ts）	-	反映文章时效性，新文章通常更受欢迎，但不同用户对时效性偏好不同



(a) 文章字数分布　　(b) 文章类别分布

图 1: 文章基础特征分布

关键发现：从分布图可以看出，文章字数主要集中在 100-300 字之间，符合新闻文章的特点。正负样本的字数分布相似，说明文章长度需要与其他特征配合使用。不同类别的文章数量差异很大，某些类别的正样本比例明显高于其他类别，说明类别特征很重要。

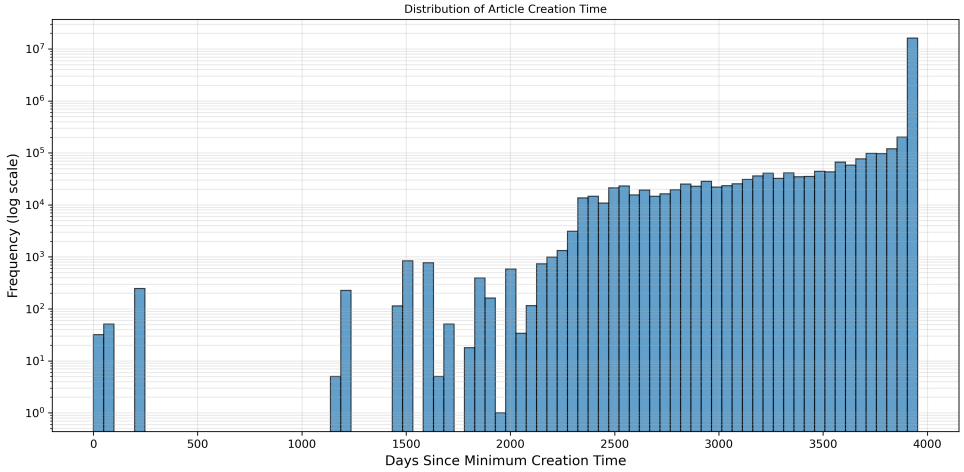
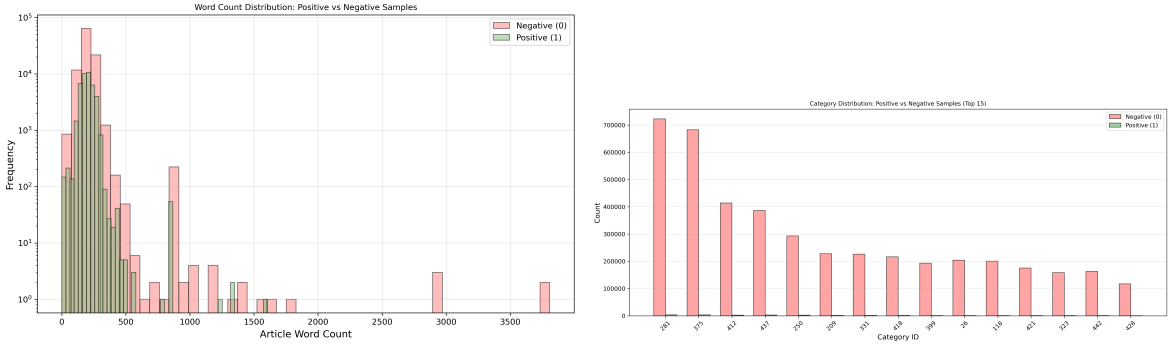


图 2: 文章创建时间分布



(a) 文章字数正负样本对比　　(b) 文章类别正负样本对比

图 3: 文章基础特征在正负样本上的分布对比

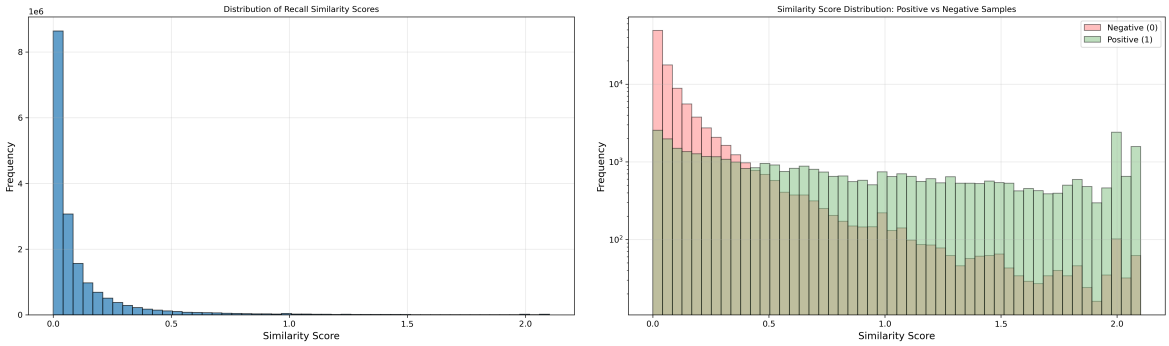
3.2.2 召回相似度特征

召回相似度特征是我们模型的核心，它们综合了多种召回方法的结果，反映文章与用户兴趣的匹配程度。这些特征的重要性最高，是模型判断用户是否会点击文章的关键依据。

表 4: 召回相似度特征总结

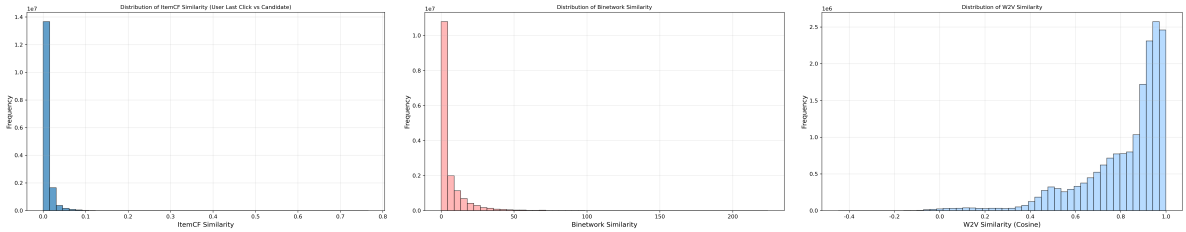
特征名称	统计值	业务含义
召回相似度分数（sim_score）	均值 0.1278，中位数 0.0442	**最重要的特征**，综合 ItemCF、Binetwork、W2V 三种方法，反映文章与用户历史兴趣的相似度。正样本平均相似度 0.8901，负样本 0.1206，差异明显
用户最后点击文章 ItemCF 相似度	-	捕捉用户**最近兴趣**，反映候选文章与用户最近阅读内容的匹配度
用户历史点击文章 ItemCF 加权相似度之和	-	反映用户**整体兴趣**，考虑所有历史点击，近期文章权重更高（0.7 幂次衰减）
用户最后点击文章 Binetwork 相似度	-	基于用户-文章网络结构，与 ItemCF 互补，能发现 ItemCF 发现不了的关联
用户最后点击文章 W2V 相似度	-	捕捉**语义相似性**，发现内容相似但用户群体不同的文章
用户最近 2 篇点击文章 W2V 相似度之和	-	综合反映候选文章与用户最近阅读内容的语义匹配程度

关键发现：相似度分数呈现明显的长尾分布，大部分文章相似度较低，只有少数文章相似度很高。**正样本的平均相似度分数（0.8901）远高于负样本（0.1206）**，说明相似度分数是强信号，验证了召回阶段的有效性。



(a) 相似度分数分布　　(b) 正负样本对比

图 4: 召回相似度特征分析



(a) ItemCF 相似度分布　(b) Binetwork 相似度分布　(c) W2V 相似度分布

图 5: 不同相似度特征分布比较

3.2.3 用户行为特征

用户行为特征反映用户的活跃度、偏好和阅读习惯，帮助我们理解不同用户的行为模式。

表 5: 用户统计特征

特征名称	业务含义
用户点击文章数量（user_id_cnt）	反映用户活跃程度，活跃用户和沉默用户的点击行为模式不同
文章被点击次数（article_id_cnt）	反映文章热度，热门文章更容易被点击，但需要平衡热门度和多样性
用户-类别点击次数（user_id_category_id_cnt）	反映用户对特定类别的偏好程度，比单纯类别特征更精细

时间相关特征：新闻推荐中时间因素非常重要，我们构建了多个时间特征来捕捉用户的时效性偏好。

表 6: 时间相关特征

特征名称	业务含义
候选文章创建时间与用户最后点击时间的差值	反映文章新鲜度，时间差小说明文章刚发布，时间差大可能已过时
候选文章与用户最后点击文章创建时间的差值	反映候选文章与用户最近阅读文章的新鲜度对比
用户连续两次点击的平均时间间隔	反映用户阅读节奏，帮助判断现在推荐文章是否合适
用户点击时间与文章创建时间的差值（均值/标准差）	均值反映用户通常看多新的文章，标准差反映偏好稳定性
用户点击时间的小时标准差	反映阅读时间规律性，标准差小说明阅读时间集中

文章内容特征：这些特征反映用户对文章内容属性的偏好。

表 7: 文章内容特征

特征名称	业务含义
用户历史点击文章的平均字数	反映用户对文章长度的长期偏好
用户最后点击文章的字数	反映用户对文章长度的最近偏好，与平均字数配合捕捉长期和短期变化
候选文章字数与用户最后点击文章字数的差值	反映文章长度变化幅度，帮助判断用户是否能接受长度变化

关键发现：字数差值主要集中在 0 附近，说明大部分候选文章的长度和用户最近看的文章长度相近，符合用户阅读习惯：用户通常会在相似长度的文章之间切换。

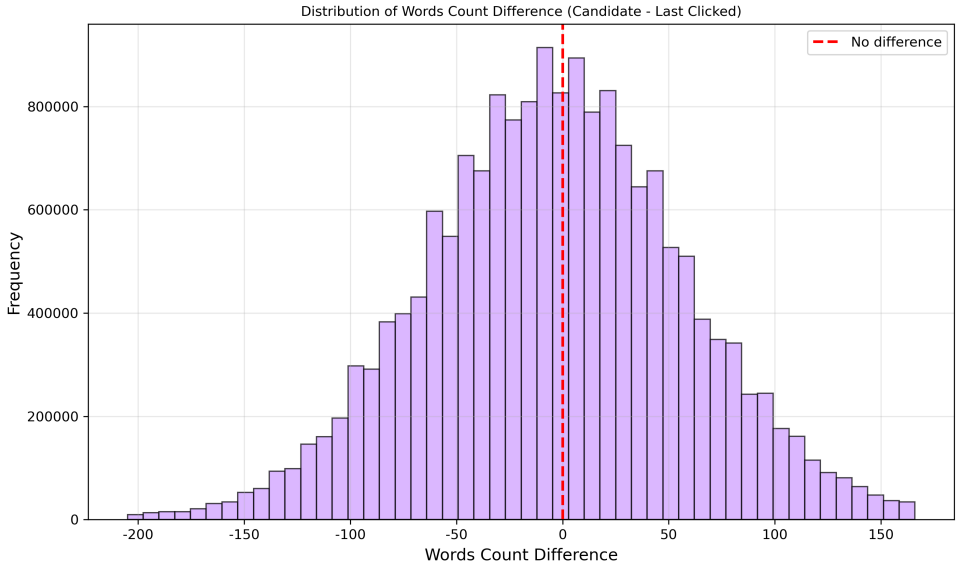
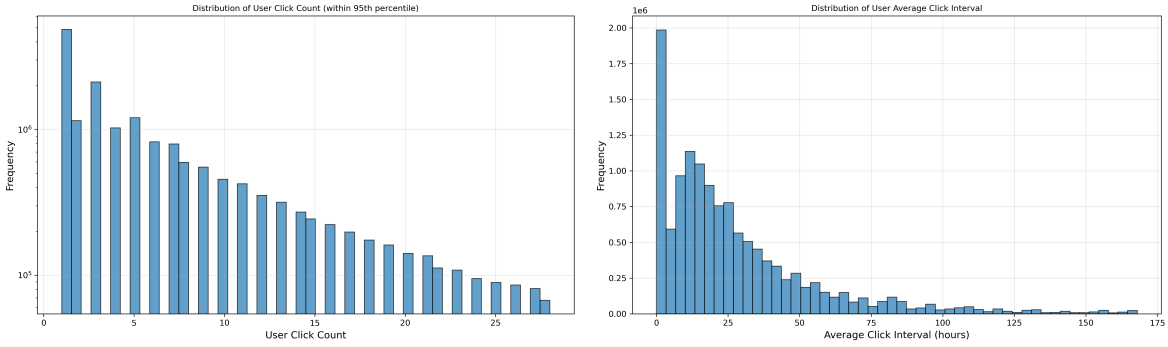


图 6: 候选文章与最近点击文章字数差值分布



(a) 用户点击数量分布　　(b) 用户点击时间间隔

图 7: 用户行为特征分析

3.2.4 标签分布与特征相关性

正负样本分布：数据分布严重不平衡，正样本（用户点击的文章）只占 0.23%，负样本占 99.77%，比例接近 1:200。这种不平衡在推荐系统中很常见，因为召回阶段会给每个用户召回很多候选文章，但用户只会点击其中的一小部分。我们通过设置正样本权重（scale_pos_weight=10）来处理这种不平衡。

表 8: 正负样本分布

样本类型	数量	比例
正样本	40,884	0.23%
负样本	8,009,580	99.77%
总计	**8,050,464**	**100%**

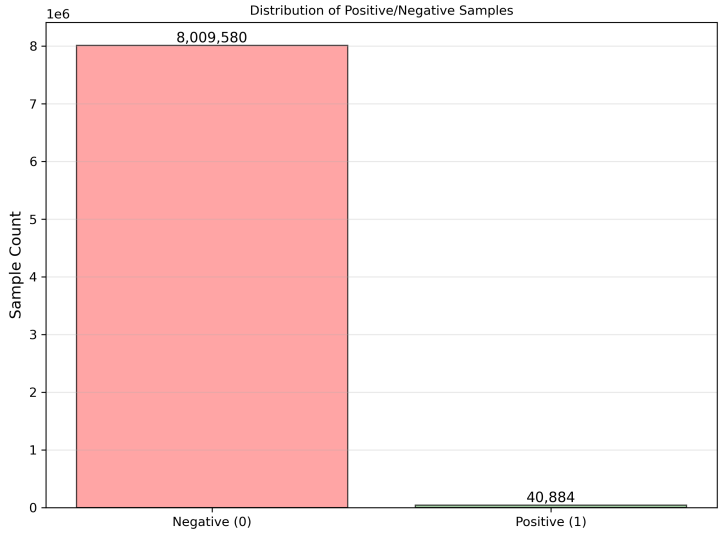


图 8: 正负样本数量分布

特征相关性分析：从特征相关性热力图可以看出：

- **相似度特征之间相关性较高：**ItemCF、Binetwork、W2V 等相似度特征存在一定相关性，但可以互补
- **时间特征相关性较低：**时间相关特征与其他特征相关性低，提供了独立信息
- **用户行为特征相关性适中：**与相似度特征有一定相关性，但提供了额外信息

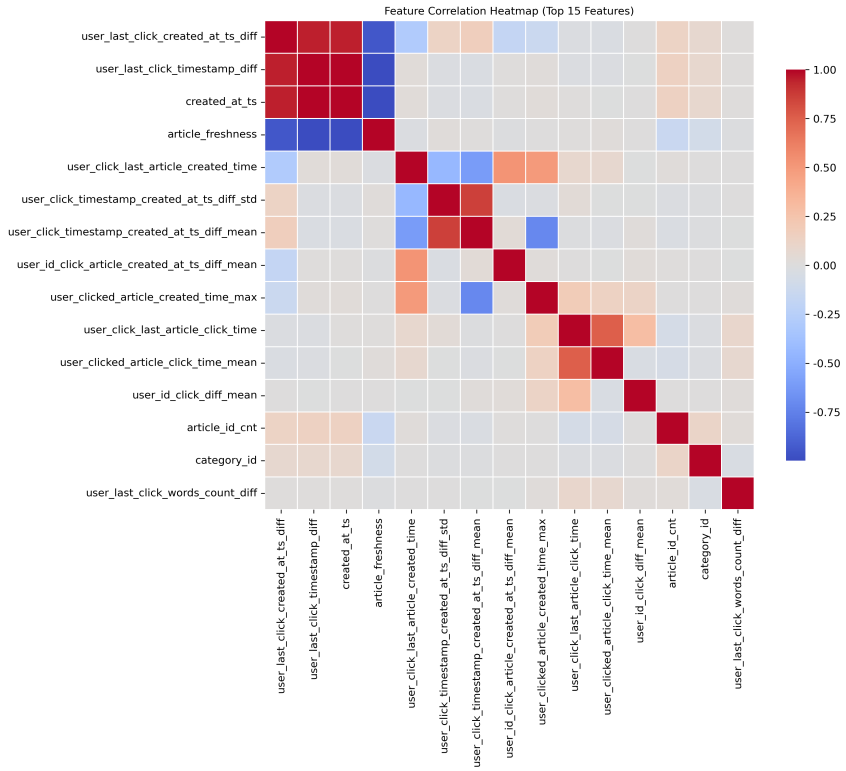


图 9: 特征相关性热力图

4 模型设计

4.1 召回阶段

我们采用多路召回策略，融合三种召回方法，从不同角度发现用户可能感兴趣的文章：

表 9: 召回方法对比

召回方法	原理	HitRate@50	MRR@50
ItemCF（基于物品的协同过滤）	计算物品之间的相似度，推荐与用户历史点击物品相似的文章	0.70	0.238
Binetwork（二分网络）	基于用户-物品二分网络结构，计算物品相似度	0.61	0.238
Word2Vec（词向量）	将用户点击序列视为句子，训练 Word2Vec 模型，计算文章相似度	0.31	0.083

召回融合策略：

- 对每种召回方法的相似度分数进行 Min-Max 标准化（将不同方法的分数统一到 0-1 范围）
- 加权融合：`sim_score = itemcf_score + binetwork_score + w2v_score`
- **最终召回指标**：HitRate@50=0.70, MRR@50=0.261（融合后性能优于单一方法）

4.2 排序阶段

排序学习（Learning to Rank, LTR）是信息检索和推荐系统中的核心问题，目标是根据查询（用户）对文档（物品）的相关性进行排序。根据训练数据的标注方式，排序学习可以分为三类：

1. Pointwise 方法

- **原理**：将排序问题转化为分类或回归问题，对每个样本独立预测其相关性分数
- **优点**：实现简单，可以使用成熟的分类/回归算法
- **缺点**：忽略了样本之间的相对顺序关系，无法直接优化排序指标
- **代表算法**：LightGBM 分类器、逻辑回归、SVM 等

2. Pairwise 方法

- **原理**：将排序问题转化为样本对的二分类问题，学习判断两个样本的相对顺序
- **优点**：考虑了样本之间的相对关系，更符合排序的本质
- **缺点**：样本对数量庞大（$O(n^2)$），训练复杂度高
- **代表算法**：RankNet、LambdaRank 等

3. Listwise 方法

- **原理**：直接优化整个列表的排序质量，使用排序指标（如 NDCG）作为损失函数
- **优点**：直接优化排序指标，理论上性能最优
- **缺点**：实现复杂，计算开销大
- **代表算法**：LambdaMART、ListNet 等

我们实现并对比了两种排序方法:Pointwise（LightGBM）和 Listwise（LambdaMART）。

4.2.1 LightGBM（Pointwise）

理论原理：

LightGBM 是一个基于梯度提升决策树（GBDT）的框架，我们将其作为 Pointwise 排序方法使用：

1. **Pointwise 排序**：将排序问题转化为二分类问题
   - 正样本：用户点击的文章（label=1）

- 负样本：用户未点击的文章（label=0）
- 模型输出：文章被点击的概率

2. **LightGBM 优势：**
   - **直方图算法**：将连续特征离散化，减少内存和计算开销
   - **Leaf-wise 生长**：只分裂增益最大的叶子节点，相比 Level-wise 更高效
   - **特征并行和数据并行**：支持分布式训练
   - **类别特征支持**：原生支持类别特征，无需 one-hot 编码
3. **缺点**：无法直接优化排序指标（如 NDCG），只能间接通过分类准确率优化

实际配置：

Listing 1: LightGBM 配置参数

```
lgb.LGBMClassifier(
    num_leaves=64,              # 叶子节点数，控制模型复杂度
    max_depth=10,               # 树的最大深度，防止过拟合
    learning_rate=0.05,         # 学习率，控制每棵树的贡献
    n_estimators=10000,         # 最大迭代次数
    subsample=0.8,              # 行采样比例，防止过拟合
    feature_fraction=0.8,        # 特征采样比例，增加随机性
    reg_alpha=0.5,              # L1正则化
    reg_lambda=0.5,             # L2正则化
    scale_pos_weight=10,        # 正样本权重，处理类别不平衡（1:196）
    min_child_samples=100,      # 叶子节点最小样本数，防止过拟合
)
```

配置说明：

- **scale_pos_weight=10**：由于正负样本比例为 1:196，设置正样本权重为 10，平衡样本分布
- **早停机制**：使用 `early_stopping(stopping_rounds=100)`，验证集性能不再提升时停止训练
- **交叉验证**：使用 `GroupKFold(n_splits=5)` 按用户分组，避免同一用户出现在训练集和验证集

特征重要性 Top 10：

表 10: LightGBM 特征重要性 Top 10

排名	特征名称	重要性分数
1	sim_score	767,792
2	user_last_click_timestamp_diff	237,118
3	user_last_click_article_itemcf_sim	155,210
4	user_last_click_article_w2v_sim	91,132
5	article_id_cnt	79,432
6	user_clicked_article_itemcf_sim_sum	69,579
7	user_last_click_created_at_ts_diff	53,313
8	user_last_click_article_binetwork_sim	43,189
9	user_click_timestamp_created_at_ts_diff_mean	29,134
10	user_click_article_w2w_sim_sum_2	24,231

训练结果（5 折交叉验证）：

表 11: LightGBM 训练结果

指标	@5	@10	@20	@40	@50
HitRate	0.46	0.59	0.70	0.77	0.78
MRR	0.281	0.298	0.306	0.309	0.309

训练时间：每个 fold 约 76 秒，总计约 6.3 分钟

4.2.2 方 LambdaMART（Listwise）

理论原理：

LambdaMART 是结合 LambdaRank 和 MART 的排序算法，属于 Listwise 方法：

1. **LambdaRank 核心思想：**
   - 定义每个样本的 Lambda 梯度，直接优化 NDCG 等排序指标
   - Lambda 值反映交换两个样本位置后 NDCG 的变化
   - 对于相关文档，如果排在无关文档后面，Lambda 会增大，促使模型提升其分数
2. **MART 实现：**
   - 使用梯度提升树拟合 Lambda 梯度
   - 每棵树预测 Lambda 值，多棵树累加得到最终排序分数
   - 通过梯度提升逐步优化排序性能
3. **优势：**
   - **直接优化排序指标：**相比 Pointwise 方法，直接优化 NDCG，理论上性能更优

- **考虑列表上下文**：能够利用列表级别的信息（如列表大小、多样性等）
- **梯度提升优势**：继承了 GBDT 的优点（特征交互、非线性建模）

4. **缺点**：实现复杂，需要 group 信息；训练时间可能较长
5. **适用场景**：对排序性能要求高、有列表级别特征的场景

实际配置：

Listing 2: LambdaMART 配置参数

```
lgb.LGBMRanker(
    num_leaves=128,                 # 更大的叶子数，增加模型容量
    max_depth=12,                   # 更深的树，捕捉复杂特征交互
    learning_rate=0.03,             # 更小的学习率，更稳定的训练
    n_estimators=15000,             # 更多迭代次数
    subsample=0.85,                 # 行采样
    feature_fraction=0.85,          # 特征采样
    reg_alpha=0.2,                  # 较小的正则化，允许更复杂的模型
    reg_lambda=0.2,
    objective='lambdarank',         # 使用lambdarank目标函数
    metric='ndcg',                  # 评估指标为NDCG
    ndcg_eval_at=[5, 10, 20, 50], # NDCG计算位置
    lambdarank_truncation_level=30, # Lambda计算时的截断级别
)
```

配置说明：

- **objective='lambdarank'**：使用 LambdaRank 目标函数，直接优化 NDCG
- **group 信息**：需要提供每个用户的样本数量，用于计算 NDCG
- **lambdarank_truncation_level=30**：只考虑前 30 个位置的样本对，减少计算量
- **更大的模型容量**：相比 LGB，使用更多的叶子节点和更深的树，以捕捉复杂的排序模式

特征重要性 Top 10：

表 12: LambdaMART 特征重要性 Top 10

排名	特征名称	重要性分数
1	sim_score_diff_from_avg	952,547
2	sim_score	427,483
3	user_last_click_timestamp_diff	278,883
4	user_last_click_article_w2v_sim	127,883
5	article_id_cnt	88,222
6	list_rank	72,591
7	user_last_click_created_at_ts_diff	50,636
8	user_clicked_article_itemcf_sim_sum	46,088
9	user_last_click_article_itemcf_sim	36,868
10	user_last_click_article_binetwork_sim	28,220

训练结果（5 折交叉验证）：

表 13: LambdaMART 训练结果

指标	@5	@10	@20	@40	@50
HitRate	0.46	0.59	0.70	0.77	0.78
MRR	0.282	0.300	0.307	0.310	0.310

训练时间：每个 fold 约 35-40 秒，总计约 3.2 分钟

列表级别特征分析：

LambdaMART 能够有效利用列表级别特征，这些特征是 Listwise 方法独有的，Pointwise 方法很难利用。这些特征帮助模型在列表内部进行比较，找到真正相对更好的文章。

表 14: 列表级别特征分析

特征名称	重要性	业务含义
sim_score_diff_from_avg	952,547	**最重要的列表特征**。反映文章在候选列表中的相对位置（相对相似度），而非绝对相似度。因为不同用户的候选列表质量不同，相对位置比绝对位置更重要。让模型能在列表内部比较，找到真正相对更好的文章。
list_rank	72,591	反映召回阶段对文章的置信度。排名越靠前，说明召回算法认为越相关。让排序模型能参考召回阶段的判断。
sim_score_diff_from_max	24,062	反映候选文章与最佳候选的差距。帮助判断文章在列表中的相对质量，差距太大可能不值得推荐。

4.2.3 模型对比分析

实际性能对比：

表 15: LightGBM vs LambdaMART 实际性能对比

指标	LightGBM	LambdaMART	差异
MRR@5	0.281	0.282	+0.001
MRR@10	0.298	0.300	+0.002
MRR@20	0.306	0.307	+0.001
MRR@50	0.309	0.310	+0.001
训练时间	6.3 分钟	3.2 分钟	-49%

性能差异分析：

1. **排序性能**：LambdaMART 略优于 LightGBM（+0.1-0.2%），符合理论预期
   - **原因分析**：LambdaMART 直接优化 NDCG，而 LightGBM 只能间接优化
   - **提升幅度小**：可能因为特征工程已经很好，两种方法都能学到有效的排序模式
2. **训练速度**：LambdaMART 反而更快（-49%），与理论预期不符
   - **可能原因**：
     - LambdaMART 使用了 `lambdarank_truncation_level=30`，只计算前 30 个位置的 Lambda，减少了计算量
     - LightGBM 需要计算所有样本的概率，而 LambdaMART 只需要计算排序分数
     - 数据分布可能使得 LambdaMART 的梯度计算更高效

3. **特征重要性差异：**
   - **LightGBM：**最关注 `sim_score`（绝对相似度）
   - **LambdaMART：**最关注 `sim_score_diff_from_avg`（相对相似度）
   - **结论：**LambdaMART 更善于利用列表上下文信息，体现了 Listwise 方法的优势

   结论：

1. **性能对比：**LambdaMART 在各项指标上略优于 LightGBM，符合理论预期（直接优化排序指标）
2. **训练效率：**LambdaMART 训练速度更快，超出理论预期（可能由于截断级别优化）
3. **特征利用：**LambdaMART 更关注列表级别特征（如 sim_score_diff_from_avg、list_rank），体现了 listwise 方法的优势
4. **最终选择：**由于 LambdaMART 性能略优且训练更快，我们选择 LambdaMART 作为最终模型
5. **理论验证：**实际结果验证了 Listwise 方法在排序任务上的理论优势，虽然提升幅度较小，但在大规模推荐系统中，即使是 0.1% 的提升也具有重要价值

5 训练

5.1 训练流程

训练流程包括数据预处理、召回、特征工程、模型训练和预测五个阶段，总耗时约 14-17 分钟。

表 16: 训练流程与时间统计

阶段	耗时	主要内容
数据预处理	约 2 分钟	加载点击日志、划分验证集（50,000 用户）、数据清洗和格式转换
召回阶段	约 3 分钟	ItemCF 召回（82 秒）、Binetwork 召回（23 秒）、W2V 召回（23 秒）、召回融合（47 秒）
特征工程	约 5 分钟	基础特征、相似度特征、时间特征、Listwise 特征（共 30 个特征）
模型训练	3-6 分钟	LightGBM（5 折，6.3 分钟）或 LambdaMART（5 折，3.2 分钟）
预测和提交	约 1 分钟	生成预测结果、格式转换和保存
总计	**14-17 分钟**	**离线验证模式**

5.2 关键优化点

我们在训练过程中采用了多项优化策略，显著提升了训练效率和模型性能：

表 17: 关键优化策略

优化策略	效果说明
向量化计算	将相似度特征计算从 apply 改为向量化，速度提升 10-50 倍
交叉验证	使用 GroupKFold 按用户划分，避免同一用户出现在训练集和验证集，防止数据泄露
早停机制	设置 early_stopping（stopping_rounds=100），验证集性能不再提升时停止训练，避免过拟合
样本权重	使用 scale_pos_weight=10 处理类别不平衡（正负样本比例 1:196）
特征缓存	保存特征文件，避免重复计算，提升迭代效率

6 比赛排名

成绩 0.2911，排名第 307，网站图片如下：

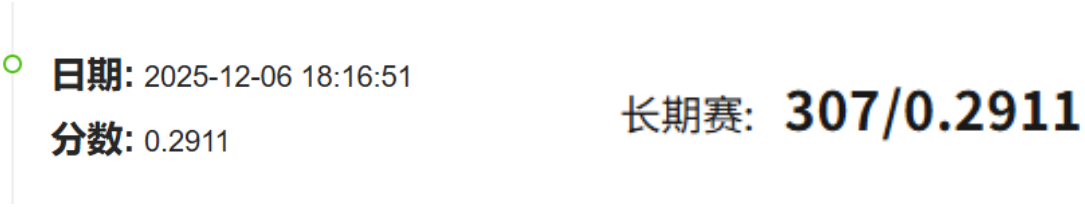


图 10: 比赛排名

7 团队成员分工

表 18: 团队成员分工

成员	主要职责
李丁	特征提取、召回算法实现与优化、实验报告撰写
罗浩民	排序模型设计与优化、实验报告撰写

8 个人总结和感悟

8.1 技术收获

1. **推荐系统架构**：深入理解了” 召回 + 排序” 两阶段推荐系统的设计和实现
2. **特征工程**：学会了如何从用户行为数据中提取有效特征，特别是时间序列特征和交互特征
3. **模型对比**：通过对比 Pointwise 和 Listwise 方法，理解了不同排序学习范式的优缺点
4. **性能优化**：掌握了向量化计算、并行处理等优化技巧

8.2 经验教训

1. **迭代优化的重要性**：通过多次迭代，逐步提升模型性能，每次迭代都有明确的改进目标
2. **特征工程是关键**：好的特征比复杂的模型更重要，sim_score 等召回特征贡献了最大的重要性
3. **处理不平衡数据**：类别不平衡是推荐系统的常见问题，需要合理设置样本权重
4. **计算效率优化**：向量化计算可以大幅提升特征工程效率，值得投入时间优化

8.3 未来改进方向

1. **深度学习方法**：尝试使用深度神经网络（如 DeepFM、Wide&Deep）进一步提升性能
2. **实时特征**：增加实时特征，如用户当前会话信息、实时热门文章等
3. **多目标优化**：不仅优化点击率，还可以考虑多样性、新颖性等指标
4. **模型融合**：尝试融合多个模型（如 LGB + LambdaMART + 深度学习模型）进一步提升性能