舆情推送内容相似度判定与去重优化

执行计划：舆情推送内容相似度判定与去重优化

背景 (Background)

当前舆情推送系统在处理大量舆情数据时，经常出现内容高度相似或重复的推送情况，导致用户收到冗余信息，降低了用户体验和系统效率。系统目前缺乏对舆情内容相似度的有效判断机制，无法在推送前对内容进行去重或合并处理。

当前状态：舆情推送系统按原始数据直接推送，未经相似度分析和去重处理，导致用户接收到大量相似内容。

期望状态：系统能够自动计算舆情内容间的相似度，标记高相似度内容，并在推送环节进行智能去重或合并，确保用户接收到高质量、低冗余的舆情信息。

目标 (Goal)

- 建立舆情内容的相似度计算机制，对即将推送的舆情内容进行相似性分析，实现高相似内容的判别，验收标准：相似度计算准确率达到90%以上
- 在舆情推送数据结构中，增加"相似度分值"字段或其他字段方式，明确标识每条内容与历史推送/当前待推送内容的相似度，验收标准：数据结构修改完成并能正确存储相似度信息
- 支持后续在推送环节根据相似度字段进行过滤、去重或合并推送，避免重复推送高度相似的内容，验收标准：重复内容推送率降低80%以上
- 完成时间：2025年8月15日

当前项目环境 (Current Project Environment)

- **技术栈**:
  - 技术栈：Python 3.9+, Django 4.2+, 已经存在的知识库输入和查询API接口
- **相关目录结构**:

- 放在notebook下面生成一个sentiment-push-content-similarity-detection-and-deduplication.py文件

实施步骤 (Implementation Steps)

☐ AI: 1. 设计并实现基于向量嵌入的文本相似度计算方案

任务描述 : 使用DashScope的文本嵌入服务和Qdrant向量数据库，实现舆情内容的向量化和相似度计算功能。

预期结果 : 能够将文本转换为高维向量，并计算不同文本间的相似度分数。

验证方法 : 使用样例数据测试相似度计算的准确性和效率。

文件路径 : notebook/sentiment-push-content-similarity-detection-and-deduplication.py

代码/命令 :

获取文本的嵌入向量

```
def get_embedding(text):
    response = client_openai.embeddings.create(
        model="text-embedding-v3", # 使用DashScope的text-embedding-v3模型
        input=text,
        dimensions=vector_size, # 指定向量维度
        encoding_format="float" # 指定编码格式
    )
    return response.data[0].embedding
```

执行结果 : 成功实现了基于DashScope的text-embedding-v3模型的文本向量化功能，该模型输出1024维的文本向量，能够有效捕捉文本的语义信息。

☐ AI: 2. 构建向量数据库存储与检索系统

任务描述 : 使用Qdrant向量数据库构建高效的向量存储和检索系统，支持相似度计算和查询。

预期结果 : 能够将文本向量存入数据库，并进行高效的相似度搜索。

验证方法 : 测试向量存储和检索的性能与准确性。

文件路径 : notebook/sentiment-push-content-similarity-detection-and-deduplication.py

代码/命令 :

创建集合

```
client.create_collection(
```

```
    collection_name=collection_name,
    vectors_config=VectorParams(size=vector_size, distance=Distance.COSINE),
)
```

添加向量到集合

```
point = PointStruct(
    id=0,
    vector=embedding1,
    payload={"text": text_src, "type": "原始文本"}
)

client.upsert(
    collection_name=collection_name,
    points=[point]
)
```

相似度搜索

```
search_results = client.search(
    collection_name=collection_name,
    query_vector=embedding2,
    limit=1  # 返回最相似的结果
)
```

执行结果 :成功实现了基于DashScope的text-embedding-v3模型的文本向量化功能，该模型输出1024维的文本向量，能够有效捕捉文本的语义信息。

▢ AI: 3. 开发样本数据处理与测试框架

任务描述 : 开发一个框架，用于加载和处理样本数据，测试相似度计算的效果。

预期结果 : 能够批量处理样本数据，并输出相似度计算结果。

验证方法 : 使用多组样例数据测试系统的稳定性和准确性。

文件路径 : notebook/sentiment-push-content-similarity-detection-and-deduplication.py

代码/命令：

读取sample.json文件

```
def load_sample_data(file_path):
    with open(file_path, 'r', encoding='utf-8') as f:
        data = json.load(f)
    return data
```

主函数处理逻辑

```
def main():
    # 加载sample.json数据
    sample_data_path = os.path.join(os.path.dirname(__file__), '舆情推送内容相似度判定与去重优化-所有的样例对比.json')
    sample_data = load_sample_data(sample_data_path)
    for key in sample_data.keys():
        # 处理每组测试数据
        # ...
```

执行结果：成功实现了样本数据的加载和处理框架，能够从JSON文件中读取测试数据，并对每组数据进行相似度计算和评估。

涉及的文件 (Involved Files)

notebook/sentiment-push-content-similarity-detection-and-deduplication.py

notebook/舆情推送内容相似度判定与去重优化-所有的样例对比.json

回滚计划 (Rollback Plan)

如果修复导致问题，按以下步骤回滚：

1. **立即回滚代码**：

```
代码块
```

```
git revert <commit-hash>
git push origin feature/sentiment-push-content-similarity-detection-and-deduplication
```

2. **验证回滚**：
   - 测试编辑非第一条消息的功能仍然正常
   - 确认系统整体功能未受影响
   - 通知用户临时不要编辑第一条消息
3. **问题分析**：
   - 分析回滚原因
   - 修复问题后重新部署
   - 添加更全面的测试覆盖