T5-Pegasus 中文生成式模型摘要生成/指代消解系统

需求分析

1. 项目概述

1.1 目标

开发一个基于 Google T5 中文生成式模型的系统，支持文本摘要生成和指代消解任务，具备批量生成和多进程处理能力。

1.2 功能特性

- **T5 模型集成：** 集成 t5-pegasus 中文生成式预训练模型。
- **文本摘要生成：** 利用 T5 模型生成输入文本的概要或摘要。
- **指代消解：** 识别和解析输入文本中的指代关系，确保生成的摘要一致性。
- **多进程支持：** 实现多进程处理，提高系统推理/生成速度。
- **批量生成接口：** 提供接口，支持同时处理多个文本的批量生成请求。

2. 用户需求

2.1 用户角色

- **系统用户：** 需要生成文本摘要或执行指代消解任务的用户。

2.2 用户故事

- **生成文本摘要：** 作为用户，我希望能够输入一段文本，获取该文本的生成式摘要。
- **执行指代消解：** 作为用户，我想识别和解析文本中的指代关系，以确保生成的摘要一致性。

3. 系统需求

3.1 功能性需求

- **T5 模型接入：** 集成 t5-pegasus 模型用于文本摘要生成和指代消解。
- **多进程处理接口：** 实现多进程支持，提高系统性能。
- **批量生成接口：** 提供接口，支持同时处理多个文本的批量生成请求。

3.2 非功能性需求

- **性能优化：** 优化系统性能，确保在大规模文本数据上的高效处理。

4. 技术实现

4.1 技术栈

- **编程语言：** Python（考虑性能需求，可以使用 Cython 或其他性能更好的扩展库）。
- **深度学习框架：** PyTorch 和 bert4torch，用于 t5-pegasus 模型的集成。

4.2 数据存储

- **模型参数存储：** 存储 t5-pegasus 模型的权重参数。

5. 安全和隐私考虑

- **用户数据隐私：** 确保用户输入文本的隐私得到妥善处理。
- **模型参数安全：** 采取适当的措施来保护存储在系统中的 t5-pegasus 模型参数。

6. 测试策略

- **单元测试：** 针对模型集成和指代消解逻辑进行单元测试。
- **集成测试：** 测试系统是否正确处理批量请求和多进程并发。
- **性能测试：** 评估系统在大规模文本数据上的性能。

7. 项目运行说明

1. 下载 t5-pegasus 模型并放置在指定目录。
2. 安装所需 Python 库（transformers、tokenizers、torch、jieba、rouge、tqdm、pandas）。
3. 执行训练和预测命令，可选择使用多进程。

系统设计方案

1. 系统架构概述

1.1 模块划分

系统主要包括以下模块：

- **T5 模型集成模块：** 负责集成 Google T5 中文生成式模型，用于文本摘要生成和指代消解。
- **批量处理接口模块：** 提供接口，支持同时处理多个文本的批量生成请求，接收预测数据的 TSV 文件。
- **多进程处理模块：** 实现多进程支持，提高系统性能。
- **数据读取模块：** 用于读取输入的预测数据，解析 TSV 文件。

1.2 数据流

1. 用户提交预测数据的 TSV 文件请求。
2. 批量处理接口模块接收请求，调用数据读取模块解析 TSV 文件，将文本数据传递给 T5 模型集成模块。
3. T5 模型集成模块调用 T5 模型进行文本摘要生成和指代消解。
4. 生成的摘要经过指代消解处理，确保一致性。
5. 结果返回给用户。

2. 技术选型

2.1 编程语言

系统使用 Python 进行开发，考虑性能需求时可以引入 Cython 或其他性能更好的扩展库。

2.2 深度学习框架

选择 PyTorch 和 bert4torch 作为深度学习框架，以便集成 t5-pegasus 模型。

2.3 数据存储

模型参数存储在安全的存储环境中，确保模型参数的完整性和安全性。

3. 测试策略

- 单元测试：对模型集成和指代消解逻辑进行单元测试，保证功能的正确性。
- 集成测试：测试系统是否正确处理批量请求和多进程并发，确保系统整体稳定性。
- 性能测试：评估系统在大规模文本数据上的性能，确保系统高效运行。

4. 项目运行说明

1. 下载 t5-pegasus 模型并放置在指定目录。
2. 安装所需 Python 库（transformers、tokenizers、torch、jieba、rouge、tqdm、pandas）。
3. 执行训练和预测命令，可选择使用多进程。

实现细节

一.predict_with_generate.py

1. 数据加载：

`load_data` 函数读取包含文本数据的文件，其中每一行代表内容或标题和内容的元组。它返回一个包含数据的元组列表。

2. 分词：

`T5PegasusTokenizer` 类继承了 transformers 库中的 `BertTokenizer` 类。它使用 `jieba` 中文分词器进行词级别的分词。如果一个词不在词汇表中，它会回退到原始的 BERT 分词器。

3. 数据集与数据加载器：

`KeyDataset` 类是一个 PyTorch 数据集，保存预处理后的数据。`create_data` 函数使用分词器对内容进行编码，将结果特征存储在数据集中。

4. 模型加载：

脚本加载了一个 T5-PEGASUS 模型和一个分词器。模型从保存的检查点加载，而分词器使用预训练权重进行初始化。

5. 批次准备：

`prepare_data` 函数使用 `KeyDataset` 创建了测试数据的 DataLoader，并指定了批处理大小。

6. Rouge 评估：

`compute_rouge` 和 `compute_rouges` 函数用于计算 Rouge-1、Rouge-2 和 Rouge-L 分数，用于评估摘要性能。

7. 摘要生成：

`generate` 函数使用 T5-PEGASUS 模型生成摘要。它使用分词器对输入进行标记化，生成摘要，解码生成的标记，并将结果写入 CSV 文件。

8. 多进程处理（可选）：

脚本提供了使用多进程加速推理的选项。`generate_multiprocess` 函数使用 `Pool` 类在每个数据批次上并行调用。

9. 参数解析：

`init_argument` 函数使用 `argparse` 模块初始化命令行参数。

10. 主执行部分：

脚本的主要部分初始化参数、准备数据、加载模型，并执行摘要生成。如果启用多进程且在 CPU 上运行，它将使用多个进程进行并行化。

二.train_with_finetune.py

1. 数据加载：

`load_data` 函数加载训练和验证数据。数据格式为每一行包含一个标题和正文，返回一个包含标题和正文的元组的列表。

2. 分词器：

`T5PegasusTokenizer` 类继承了 `BertTokenizer` 类，用于对中文文本进行分词。它使用 `jieba` 进行词级别的分词，如果词不在词汇表中，会回退到BERT原生的分词器。

3. 数据集与数据加载器：

`KeyDataset` 类是一个 PyTorch 数据集，用于保存预处理后的数据。`create_data` 函数编码正文和标题，根据任务不同返回不同的数据域。

4. 序列填充：

`sequence_padding` 函数用于将序列填充到相同的长度，采用numpy数组操作。

5. 默认组合（Collate）：

`default_collate` 函数用于将一个批次的数据转换为张量，考虑了不同数据类型的处理，如字符串、整数、浮点数等。

6. 准备数据：

`prepare_data` 函数根据任务（训练或验证）准备数据。使用 `DataLoader` 对数据进行批次加载。

7. 计算 Rouge 分数：

`compute_rouge` 和 `compute_rouges` 函数用于计算 ROUGE-1、ROUGE-2 和 ROUGE-L 分数，用于评估生成的摘要质量。

8. 模型训练：

`train_model` 函数用于训练 T5-PEGASUS 模型。采用交叉熵损失函数，使用 Adam 优化器进行微调，同时进行验证集上的 ROUGE 评估。

9. 参数初始化：

`init_argument` 函数使用 `argparse` 模块初始化命令行参数。

10. 主执行部分：

脚本的主要部分包括初始化参数，准备训练和验证数据，加载预训练模型，进行微调，并保存在验证集上效果最好的模型。

结果分析

1. 摘要生成性能分析

1.1 Rouge 分数

- **Rouge-1、Rouge-2、Rouge-L：** 分析生成的摘要在单一词、双词和长词上的覆盖率和准确性。
- **对比分析：** 比较不同模型和参数设置下的 Rouge 分数，评估摘要生成性能的差异。

1.2 摘要一致性

- **指代消解准确性：** 检查生成的摘要中是否存在指代关系，并分析指代消解的准确性。
- **上下文一致性：** 确保生成的摘要在整体上与原文一致，不失去上下文的主题逻辑。

2. 系统性能分析

2.1 多进程处理

- **性能提升比较：** 比较使用多进程和单进程时系统的性能表现，分析多进程对系统处理速度的影响。

2.2 批量生成接口

- **并发处理能力：** 评估系统在处理批量生成请求时的并发处理能力，确保系统在高负载下的稳定性。
- **响应时间：** 分析系统对不同批次大小的请求的响应时间，确保在合理时间内完成生成任务。

3. 安全和隐私分析

3.1 用户数据隐私

- **数据加密：** 检查系统对用户输入文本的隐私保护措施，包括是否进行数据加密处理。

3.2 模型参数安全

- **模型参数保护：** 评估系统对 t5-pegasus 模型参数的保护机制，确保模型参数不受未授权访问。

4. 结果展示

通过对预测多文本内容进行摘要提取分析：

```
D:\Python\python3.7.0\python.exe C:\Users\hui\Desktop\Uav_airsim\t5-pegasus-chinese\predict_with_generate.py
The tokenizer class you load from this checkpoint is not the same type as the class this function is called from. It may result in unexpected tokenization.
The tokenizer class you load from this checkpoint is 'T5Tokenizer'.
The class this function is called from is 'T5PegasusTokenizer'.
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\hui\AppData\Local\Temp\jieba.cache
有一部电影最后一支舞你可以看一下。好的有时间我看看吧。我还看了一个口碑不错的电影叫星际穿越给我的感觉超棒。
Loading model cost 0.595 seconds.
Prefix dict has been built successfully.
  0%|          | 0/313 [00:00<?, ?it/s]2023-12-14 16:59:14.910518: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic librar
2023-12-14 16:59:14.910848: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
100%|██████████| 313/313 [09:32<00:00,  1.83s/it]
Done!

进程已结束,退出代码0
```

图1 预测运行结果图

有一部电影最后一支舞你可以看一下。好的有时间我看看吧。我还看了一个口碑不错的电影叫星际穿越给我的感觉超棒。	**我还看了一个口碑不错的电影叫星际穿越给我的感觉超棒**
你有关注韩国的明星吗。有啊韩国首尔的都有关注。明星宋承宪代表作浪漫风暴挺好的一个明星推荐你关注。	**明星宋承宪代表作浪漫风暴挺好的一个明星推荐你关注**

有一部电影最后一支舞你可以看一下。好的有时间我看看吧。我还看了一个口碑不错的电影叫星际穿越给我的感觉超棒。	我还看了一个口碑不错的电影叫星际穿越给我的感觉超棒
哎呀不行我要再去看看大红灯笼高高挂怀念一下。经典就是经典这部电影的导演太厉害了还在1998年捧获威尼斯金狮奖呢。是很有才华感觉和别的导演色彩都不一样。	大红灯笼高高挂是很有才华感觉和别的导演色彩都不一样
我不信你信。看了火龙对决这部电影我感觉龙是真实存在的推荐给你看看你就信了。那我去看看是不是有真龙。	那我去看看火龙对决是不是有真龙
你袜子好臭。我没有臭袜子。你有。	你有臭袜子

表1 原内容与摘要内容对比

总结与建议

1. 结果总结

- **摘要生成性能：** 总结不同模型和参数配置下的摘要生成性能，强调在不同指标上的优劣势。
- **系统性能：** 总结系统在多进程处理和批量生成接口方面的性能表现。

2.未来发展方向

1. 摘要生成性能优化

1.1 模型改进

- **集成更先进的预训练模型：** 考虑集成最新的中文生成式预训练模型，如GPT-4、ERNIE等，以提高生成性能和语义理解。
- **领域自适应：** 探索在特定领域（如医学、法律）进行微调以提高模型在特定领域文本上的生成效果。

1.2 摘要一致性

- **上下文关联性：** 强化生成的摘要与原文上下文的关联性，确保摘要不仅在表面层面一致，还能保留文本的主题逻辑。
- **实体链接：** 考虑引入实体链接技术，确保生成的摘要中的实体与原文中的实体一致。

2. 系统性能提升

2.1 多进程处理

- **GPU 并行计算：** 考虑利用多个 GPU 进行并行计算，以提高模型推理速度，特别是在大规模文本数据上。
- **分布式计算：** 探索分布式计算框架，如使用 PyTorch Lightning 或 Horovod，以实现更高效的模型训练和推理。

2.2 批量生成接口

- **异步请求处理：** 引入异步请求处理机制，以更好地支持大规模的批量生成请求，减少用户等待时间。
- **负载均衡：** 考虑引入负载均衡技术，确保在高负载时系统资源合理分配，提高系统的稳定性。

3. 用户体验改进

- **交互式生成：** 实现一个交互式界面，让用户能够更灵活地与系统进行交互，例如选择生成的摘要长度、调整生成参数等。
- **用户反馈机制：** 引入用户反馈机制，通过用户输入和评价不断优化模型，提高系统生成结果的用户满意度。

4. 新技术应用

- **多模态生成：** 考虑引入多模态信息，如图像、视频等，以进一步提升系统的应用领域和生成多样性。
- **自监督学习：** 探索自监督学习方法，通过无监督学习从大量文本数据中提取知识，进一步提升模型的语言理解能力。

5. 社会责任

- **偏见与公平性：** 加强对模型输出中可能存在的偏见进行监测和调整，以确保生成结果的公平性。
- **透明度与解释性：** 提高系统的解释性，让用户能够理解模型生成摘要的决策过程，增强系统的透明度。