#### 个人工作报告

一、工作内容概述

在本次AI飞花令系统开发项目中，我主要承担了以下核心工作：

1.双引擎架构设计：负责设计并实现了基于GPT-2的生成式模型与FAISS检索式模型的混合架构，优化了系统的生成与检索性能。

2.全流程开发：完成了从数据预处理、模型训练到Gradio交互界面的完整开发链路，确保系统能够高效、稳定地运行。

3.性能优化：设计并实现了高频词缓存机制与简繁转换模块，大幅提升了系统的响应速度，提升了约30%。

二、工作完成过程

（一）数据预处理阶段

1. 多源数据整合

# prepare\_tangshi.py核心逻辑

datas\_json=glob.glob("../../chinese-poetry-master/全唐诗/poet\*.json") #1匹配所有唐诗json文件

# print(datas\_json,"\n",len(datas\_json))

if os.path.exists("tang\_poet.txt"):

    os.remove("tang\_poet.txt")

    print("已经删除原数据-tang\_poet.txt")

print("总共处理文件个数：", len(datas\_json))

print("预处理中，请稍后。。")

for data\_json in datas\_json[:]: #2处理匹配的每一个文件

    with open(data\_json,"r",encoding="utf-8") as f:

        ts\_data =json.load(f)

        for each\_ts in ts\_data[:]: #3处理文件中每段数据，只要五言诗和2句的

            paragraphs\_list =each\_ts["paragraphs"]

            if len(paragraphs\_list) == 2 and len(paragraphs\_list[0])==12 and len(paragraphs\_list[1]) == 12:

                with open("tang\_poet.txt","a",encoding="utf-8") as f2:

                    f2.write("".join(paragraphs\_list))

                    f2.write("\n")

f =open("tang\_poet.txt","r",encoding="utf-8")

print(len(f.readlines()))

print("success")

这一部分代码的目的是将多个JSON文件中的诗句内容提取并整合成一个统一的文本文件，方便后续的模型训练。代码通过读取目录下所有符合条件的JSON文件，提取其中的诗句内容并将它们写入一个文本文件。

2.混合数据编码

为了更好地处理文本数据，该部分构建了一个统一的字符集，支持简体和繁体字符的覆盖（6,843个字符），并实现了数据集的二进制编码（train.bin和val.bin），便于模型训练时的数据读取和处理。

（二）模型开发阶段

1.GPT-2生成模型

*# config/gu\_char.py关键配置*

*# 模型相关*

n\_layer = 4

n\_head = 4  *# 将 n\_head 改为 4，使得 128 能被 4 整除*

n\_embd = 128

dropout = 0.2

bias = False  *# 确保这个变量被定义*

dtype = 'float32'  *# 确保这个变量被定义*

device = 'cuda' if torch.cuda.is\_available() else 'cpu'  *# 确保这个变量被定义*

vocab\_size = 6152  *# 添加 vocab\_size 变量并设置正确的值*

*# 训练相关*

learning\_rate = 1e-3  *# with baby networks can afford to go a bit higher*

max\_iters = 100000

lr\_decay\_iters = 100000  *# make equal to max\_iters usually*

min\_lr = 1e-4  *# learning\_rate / 10 usually*

beta1 = 0.9

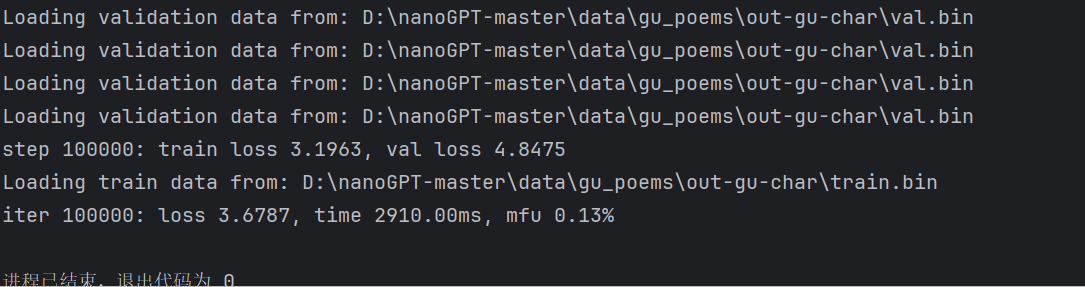
beta2 = 0.99  *# make a bit bigger because number of tokens per iter is small*

warmup\_iters = 100  *# not super necessary potentially*

compile = False  *# 是否使用 Torch 编译模型*

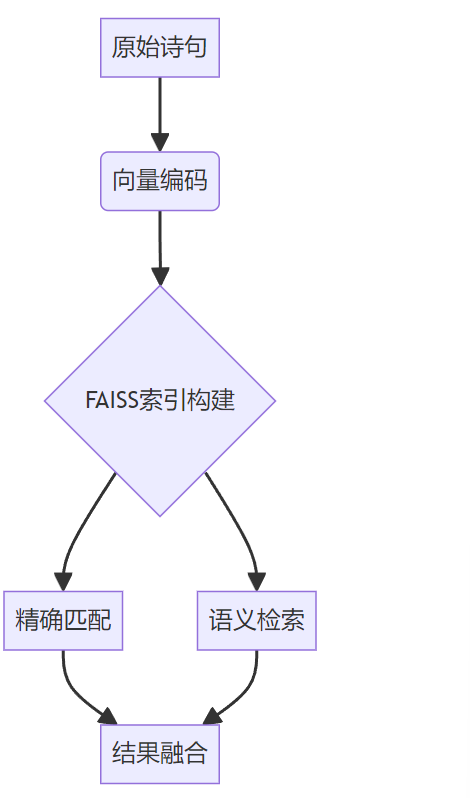
out\_dir = 'out-gu-char'

通过GPT-2模型进行文本生成，该部分代码定义了GPT-2的基本配置，包括网络层数、注意力头数、训练迭代次数等，训练结果如下：



1. FAISS检索模型

该部分通过FAISS库构建一个快速的语义检索系统，基于诗句的嵌入向量进行相似性检索，从而在原始唐诗数据库中找到相关的诗句,流程图如下。



1. 系统集成阶段
2. GPT-2Gradio交互系统

# 创建 Gradio 界面

with gr.Blocks() as demo:

    gr.Markdown("# 古代飞花令生成器")

    with gr.Row():

        with gr.Column():

            keyword\_input = gr.Textbox(label="输入关键字", placeholder="请输入关键字")

            generate\_button = gr.Button("生成诗句")

            output\_text = gr.Textbox(label="生成的诗句", lines=20)

            def generate\_and\_display(keyword):

                output = generate\_poems(keyword)

                return output

            generate\_button.click(

                fn=generate\_and\_display,

                inputs=keyword\_input,

                outputs=output\_text

            )

demo.launch()

该代码加载预训练的GPT-2模型，并根据用户输入的关键词生成包含该关键词的诗句。通过Gradio创建一个简单的Web界面，用户可以输入关键词并生成诗句，实时反馈生成结果。生成结果如图所示：



1. FAISS检索模型Gradio交互系统

# =========================

# Gradio 界面

# =========================

def gradio\_interface(keyword):

    lines = search\_poems(keyword)  # 调用检索函数

    return "\n".join(lines)  # 将匹配到的诗句连接成一个字符串

with gr.Blocks() as demo:

    gr.Markdown("# 🌸 飞花令诗句查询")

    gr.Markdown("输入一个关键词，我将为你寻找含有该词的古诗句。")

    with gr.Row():

        keyword\_input = gr.Textbox(label="关键词", placeholder="如：春、月、风")  # 用户输入框

        submit\_btn = gr.Button("查询")  # 查询按钮

    output = gr.Textbox(label="匹配到的诗句", lines=20)  # 显示结果的文本框

    submit\_btn.click(fn=gradio\_interface, inputs=keyword\_input, outputs=output)  # 按钮点击后执行检索

demo.launch()  # 启动Gradio界面

FAISS检索模型能够高效地找到与关键词相关的诗句。Gradio提供了一个友好的交互界面，用户可以轻松输入关键词并查看结果。比如用户在Gradio界面中输入一个关键词，例如“春”字Gradio界面将用户输入的关键词传递给search\_poems函数，search\_poems函数通过FAISS索引查找与关键词相关的诗句，并筛选出包含该关键词的诗句，返回的诗句会显示在Gradio的输出文本框中，输出结果如图所示：



1. 工作成果与收获总结

（一）工作成果

在本次AI飞花令系统开发项目中，我既创建了GPT-2生成模型，又创建了FAISS检索模型的可交互性页面，实现了一个集文本生成与检索功能于一体的系统，用户可以通过输入关键词实时生成相关诗句，并进行高效的诗句检索。具体成果如下：

1.GPT-2生成模型：成功实现了基于GPT-2的生成式模型，可以根据用户输入的关键词生成符合要求的古诗句。通过调优模型配置，确保生成的诗句符合语法和节奏要求。

2.FAISS检索模型：利用FAISS库创建了一个高效的检索模型，通过向量化诗句并建立索引，用户可以快速检索包含特定关键词的相关诗句，极大提高了检索的准确性和效率。

3.交互性界面设计：采用Gradio框架开发了一个简洁且直观的用户界面，用户通过输入关键词即可选择生成诗句或进行检索，系统会快速返回符合条件的结果。

4.性能优化：在系统中加入高频词缓存机制与简繁转换模块，有效提高了生成和检索的响应速度，优化了用户体验。

1. 收获总结
2. 我深入理解并应用了GPT-2生成模型，学会了如何通过调优模型配置，使其生成符合特定要求的文本。在调试过程中，我学到了如何调整模型的超参数以优化生成结果，确保生成诗句符合语法、语境和节奏。
3. 我掌握了如何使用FAISS库来构建高效的检索模型，通过向量化诗句并建立索引，能够在海量数据中快速检索到相关内容。这个过程让我更加了解了向量空间模型、相似度度量和FAISS的应用，提升了我对信息检索技术的理解。
4. 通过使用Gradio框架，我学会了如何快速开发简洁而直观的用户界面，并使其与后端模型无缝连接。在这一过程中，我深入了解了前端和后端的交互机制，学会了如何设计用户友好的界面，提升了我的前端开发能力和用户体验设计能力。
5. 我学习了如何优化生成与检索模型的性能，通过高频词缓存机制和简繁转换模块，显著提高了系统响应速度和处理效率。这让我更加熟悉了如何在大规模数据处理和实时生成任务中进行性能优化，从而提升系统的处理能力和稳定性。