期中大作业 - 文本检索

概述

本次作业主要内容是实现文本检索：给定一个文本数据集，用户输入检索词，运行程序返回与之相关的文章。同时结合网络通信，以客户端、服务器端两部分实现检索服务。

功能要求

1. 根据输入的检索词实现检索功能，支持多检索词，数量为1-3个。
2. 检索结果需要按照与检索词的相关性从高到低排序。
3. 运用C/S架构，实现客户端与服务器端的通信，以及服务器端的并发处理。文本检索的主要计算工作在服务器端完成。
4. 以简单的图形界面实现客户端与用户之间的交互，接收检索词，展示检索结果。

数据集介绍

本次作业提供了英文（all_news.csv）与中文数据集（all_cnews.csv），均为utf-8编码。作业中任选其一即可，推荐使用英文数据集完成作业。

- 英文数据集中每行为一篇新闻，包括标题（title）、正文（body）、主题（topic）、编号（id）。该数据集共有2225篇文章，分为五类，分别为business、entertainment、politics、sport、tech。
- 中文数据集中每行为一篇新闻，包括文章（text）、主题（topic）、编号（id）。该数据集共有5000篇文章，分为十类，分别为体育、财经、房产、家居、教育、科技、时尚、时政、游戏、娱乐。（中文数据集使用的是UTF-8编码）

模型要求

*该部分仅给出推荐实现方案，可以做任何可能的优化。

1. **数据处理（20'）。需完成**：读入数据集，对数据集中的文章进行去标点、分词、去除停用词及低频词，英文数据集可以考虑提取词根等。构建词典（下述检索词及相似词都只需在词典范围内完成），保存为*vocab.txt*。（该部分可直接调包）
2. **检索排序（30'）。需完成**：根据输入的检索词检索文章，并对检索到的文章进行初步排序，并返回。叙述你的排序方案的合理性。该部分需完成*C/S*架构，具体要求见下。
3. **排序优化（20'）。**对于文档-词的*TF-IDF*矩阵，我们可以将每行视为用词语的TF-IDF值表示的文章向量，对该矩阵进行降维操作，就可以得到低维的文章向量，由此我们就可以度量文章之间的相似度。更进一步我们就可以用HITS算法或求中心度等方式，取得更优的检索结果。**需完成**：构建*TF-IDF*矩阵，降维得到文章向量，并根据文章向量优化检索结果。（该部分不建议直接调用*TfidfVectorizer*，直接调用会扣一定分数）
4. **文章聚类与检索结果评测（10'）。**

   以下两点选择其一完成即可

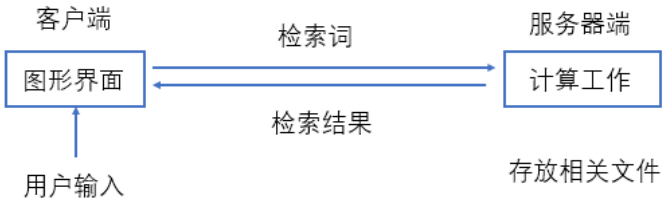
   - 文章聚类：如何验证我们得到的文章向量的合理性呢？我们可以通过对文章向量进行聚类，并与原数据集中的正确分类进行比较，计算*Purity*或*NMI*等。若你使用了不同方法得到了不同的文章向量，可以通过这种方式比较。**需完成**：文章聚类与评价。（该部分可直接调包）

    - 检索结果评测：选择一种合理的方案评价模型的检索结果。例如，要求返回的top K篇文章中包含的话题尽可能一致，而含有检索词但排在K之后的文章所包含的话题尽可能与靠前的有区分，体现出主题检索（topic search）的效果。top K具体的K值选择可以自己根据数据情况合理设计，或通过某种算法动态给出。也可以自己设计其他合理的评价方案，好的实现可以获得加分。**需完成**：模型检索结果评价。
5. **相似词（10'）。**对于输入的检索词，我们可以通过拓展相似词的方式进行模糊匹配。我们需要得到词向量来挖掘每个词的相似词，这里可以选择直接使用我们之前得到的*TF-IDF*矩阵生成词向量，也可以通过词-词共现矩阵降维得到词向量。**需完成**：挖掘相似词，为每个词保留一定数量的相似词，并保存为*synonym.txt*。（该部分不能直接调用*word2vec*生成词向量，直接调用会扣一定分数；但可以在完成基本要求后尝试）
6. **模糊匹配（10'）。需完成**：利用相似词拓展检索结果集合，并排序。

Client/Server架构

本次作业需要实现客户端和服务器端之间的socket通信，客户端负责将用户的检索请求发送到服务器端，服务器端处理完成后，将结果返回客户端。大致结构如下图所示：

- 在客户端，用户通过图形界面与文本检索系统交互
- 图形界面能够接收用户的文字输入，并展示检索结果
- 服务器端主要负责与检索相关的计算工作（找到相关文章并排序），以及存放相关文件（包括文本数据、中间结果等等）
- 服务器端需要实现合理的并发处理方案，具体采用的方式可以自由选择



`local_server.ipynb`中定义了一个简单的本地服务器。客户端和服务器端之间具体的通信内容和格式不一定要和上图一样，合理即可。

图形交互界面

Tkinter是Python GUI编程的标准库。`GUI.ipynb`中用TKinter编写了一个简单的用户图形界面框架，有余力的同学可以按照自己的喜好修改界面布局。

用户图形界面主要可分为三个部分：

- 检索主界面：用户在检索框中输入想要检索的词语，提交后开始进行检索
- 检索结果标题页：展示出与检索词相关的所有文章的标题，需要按照相关性从高到低排序
- 检索结果全文页：点击标题列表中的某个标题，显示出这篇文章的具体内容

加分项（可选，最多不超过10分）

本次作业能获得的加分上限为10分，完成以下列出的其中一项即可，多选无效。

- 对每篇文章进行关键词提取，通过检索词与文章关键词匹配，优化排序（10分）
- 优化TKinter的功能，例如增加不同检索方法的选项等。（10分）

- 在模型要求的基础上，每部分都可以尝试不同的方法完成，并比较其优劣。每部分分别计分，均可获得5分的加分，完成特别优异者可加10分。

文件说明

`local_server.ipynb`：在本机上模拟服务器端。需要实现文本检索的相关功能，与客户端的通信，以及并发处理客户端发来的请求。

`GUI.ipynb`：定义了客户端的用户图形界面，可以参照个人喜好自行设计。需要实现与服务器端的通信。

`run.ipynb`：启动客户端图形界面。在启动之前，先在local_server中运行服务器端。

`ipynb_importer.py`：ipynb文件import包的辅助文件，可以无视。

`all_news.csv`：英文数据集

`all_cnews.csv`：中文数据集（若使用中文数据集，在返回检索结果时，可先用编号替代标题）

这里提供的所有代码框架仅供参考，在做作业的过程中可以根据实际需要自由修改。

提交内容

1. 源代码：
    - 需包含注释，每个函数至少需要一条注释以说明功能。
    - 在实现文本检索算法时，如果模型比较复杂，也可在不同的notebook文件中单独实现各个部分，提交时请简单说明各个notebook文件的作用。
    - 如果有服务器端运行必需的中间计算结果，请一并提交。
2. 实验报告：需包含基础要求实现情况、加分项实现情况，描述使用的检索算法，尽量以图片的方式呈现实现的功能，以PDF的形式提交。
3. 上述文件打包为"姓名+学号+期中大作业.zip"提交，无需包含使用的数据集。