CS336-assignment4

一、look_at_cc

(a）下载上面的 WARC 文件，查看第一个网页。它的 URL 是什么？它还能访问吗？您能通过查看原始 HTML 代码判断出这个网页的内容吗？

答：它的 URL 是 http://0371rykj.com/ipfhsb/34.html，不能访问了。通过 HTML 代码大概判断这个是上海林频仪器股份有限公司试验箱产品的介绍页面。

（b）查看相应的 WET 文件，您是否认为文本中有一些部分应该被提取器过滤掉？思考一下这段文本作为训练数据的质量：使用这样的文本训练模型可能会出现什么问题？反过来，模型可以从这个页面中提取出哪些有用的信息？

答：我认为文本中开头其它网站广告文字部分应该被过滤，使用这样的文本训练的模型可能会输出黄色网站相关的文字。模型可以从这个页面提取各种试验箱的参数功能和用途。

（c）好的训练示例与上下文息息相关。请描述一下这个示例在训练数据中可能有用的应用领域，以及在训练数据中可能无用的应用领域。

答：这个示例在航空、汽车、家电、科研等领域可能有用，在艺术、音乐领域可能无用。

（d）让我们再看几个例子，以便更好地了解 Common Crawl 的内容。浏览另外 25 条 WET 记录。对于每条记录，请简要评论一下文档的语言（如果您能识别的话）、域名、页面类型等等。您需要多少个例子才能看到您认为的“高质量”网页？

第二条记录

语言：中文；域名：

http://10www.chinatikfans.com/home.php?mod=space&uid=4693&do=blog&classid=104&view=me;

页面类型：个人帖子主页。

第三条记录

语言：英文；域名：http://13.usnccm.org/；页面类型：第十三届美国全国计算力学大会官网。

第四条记录

语言：中文；域名：http://176.utchat888.com/index.phtml?PUT=gift_send&AID=220083&FID=632054；页面类型：色情网站。

第五条记录

语言：中文；域名：http://176766.cn/Article-3806781.html；页面类型：硬度计的维修与保养。

第六条记录

域名：http://178mh.com/html/,detail/,newsmovie/,wd/moAGt5YrbE/po7oqy.html,/；detail.html 模板不存在。

第七条记录

语言：中文；域名：http://1796370.tgtg97.com/?PUT=a_show&AID=266989&FID=1796370&R2=&CHANNEL=；页面类型：live173影音。

第八条记录

语言：中文；域名：http://18sex.v340.info/?&R2=&P=11&OP=&CHANNEL=；页面类型：本土自拍性爱影片。

第九条记录

语言看不懂。

第十条记录

语言看不懂。

第十一条记录

语言看不懂。

第十二条记录

语言：中文；域名：http://1s6605084.yhxzseo.com/?yibinnvfdkcl857593；页面类型：澳客竞彩比分。

第十三条记录

语言看不懂。

第十四条记录

语言：英文；域名：http://24ktcasino.net/2019/06/25/laos-casinos-4/；页面类型：Laos Casino 的介绍。

第十五条记录

404 Not Found

第十六条记录

语言：中文；域名：http://2l6185919.yizhangting.com/?shaoguan65gdkcl329611；页面类型：39健康网。

第十七条记录

语言：中文；域名：http://303323.com/xydt/dnqgqzl-65615.html；页面类型：大连鑫海合星科技有限公司官网。

第十八条记录

语言：中文；域名：http://30bad.com/edu-46963.html；页面类型：真不卡影院 电影盒子，提供免费电影的网站。

第十九条记录

语言：中文；域名：http://312001.net/home/djdt/list；页面类型：街道社区卫生服务中心主页。

第二十条记录

语言：中文；域名：http://354577.mwe075.com/index.phtml?PUT=a_show&AID=257266&FID=354577&R2=&CHANNEL=；页面类型：约爱神器。

第二十一条记录

语言：英文；域名：http://356.schoollibrary.edu.pe.ca/cgi-bin/koha/opac-search.pl?q=su:Painters%20%20and%20su:United%20States%20%20and%20su:Biography&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=branch:356&limit=su-to:Artists.&limit=su-to:Artists.&limit=su-to:Art%20appreciation.&limit=se:Getting%20to%20know%20the%20world's%20greatest%20artists&limit=su-to:Painters&limit=su-to:Artists.&limit=au:Venezia,%20Mike.&limit=se:Getting%20to%20know%20the%20world's%20greatest%20artists&limit=se:Getting%20to%20know%20the%20world's%20greatest%20artists&limit=su-to:Artists.&limit=su-to:Painting,%20American.&limit=se:Getting%20to%20know%20the%20world's%20greatest%20artists&limit=su-to:Art%20appreciation.&limit=se:Getting%20to%20know%20the%20world's%20greatest%20artists&limit=su-to:Artists.&sort_by=relevance_asc&limit=available；页面类型：PEI 学校图书挂系统目录。

第二十二条记录

语言：中文；域名：http://366392.haaxz.com/?PUT=a_show&AID=104124&FID=366392&R2=&CHANNEL=；页面类型：情色线上影片。

第二十三条记录

语言：中文；域名：http://366392.haaxz.com/?PUT=a_show&AID=130260&FID=366392&R2=&CHANNEL=；页面类型：情色线上影片。

第二十四条记录

语言：中文；域名：http://387tel.com/index.phtml?PUT=a_show&AID=244724&FID=1361125&R2=&CHANNEL=；页面类型：视讯聊天交友网。

第二十五条记录

语言看不懂。

大概每五个例子会有我认为的“高质量”网页。

二、extract_text

（a）编写一个函数，从包含原始HTML的字节字符串中提取文本。

代码块

```
def run_extract_text_from_html_bytes(html_bytes: bytes) -> str | None:
    bytes_type = detect_encoding(html_bytes)
    try:
        unicode_html = html_bytes.decode(bytes_type, errors='replace')
    except (UnicodeDecodeError, TypeError):
        unicode_html = html_bytes.decode('utf-8', errors='replace')
    text = extract_plain_text(unicode_html)
    return text
```

（b）在单个 WARC 文件上运行文本提取函数。将其输出与相应 WET 文件中的提取文本进行比较。有哪些差异和/或相似之处？哪种提取方法看起来更好？

答：拿第一条记录来看，从 WARC 提取出来的文字基本与 WET 完全相同，但是两者的排版不一样。WARC 提取出来的文字会带 · ，并且还有很多换行，文字看起来很稀疏。WET 文本看起更好。

三、language_identification

（a）编写一个函数，输入一个 Unicode 字符串，并识别该字符串中的主要语言。函数应返回一个对，包含语言标识符和 0 到 1 之间的分数，该分数表示该预测的置信度。

代码块

```
def run_identify_language(text: str) -> tuple[Any, float]:
    text = text.replace('\n', '')
    model = fasttext.load_model(str(MODEL_PATH / "lid.176.bin"))
    language, score = model.predict(text, k=1)
    return str(language[0]).split('__label__')[1], score[0]
```

（b）语言模型在推理时的行为很大程度上取决于它们所训练的数据。因此，数据过滤流程中的问题可能会导致下游问题。您认为语言识别过程中的问题可能引发哪些问题？在高风险场景中（例如部署面向用户的产品时），您将如何缓解这些问题？

答：语言识别错误可能导致的问题有：1.模型在目标语言的表现下降，如果原本打算只用英文数据训练模型，但语言识别系统把其它语言识别为英文放入训练，就会混入噪声，导致模型英文能力变差。2.多语言混淆，模型在生成时可能夹杂非目标语言。3.不一致的风格和语义偏差，不同语言的表达习惯和上下文逻辑不同，混合其它语言的文本学习可能使得模型学习到不一致的风格和价值观，影响输出稳定性。4.合规和伦理风险，某些语言的数据文本可能包含非法、敏感或不符合地区政策的内容。如果语言识别错误将这些数据标记为安全的，就有可能在下游产品中暴露敏感内容，带来法律或伦理风险。

我会在语言识别后进行检查，避免出现意外的语言，在部署时加入输出语言识别，对异常语言输出进行警告或拦截。

（c）使用从 WARC 文件中提取的文本运行语言识别系统。手动识别 20 个随机样本中的语言，并将标签与分类器预测进行比较。报告任何分类器错误。有多少文档是英语？根据观察，在过滤中使用的分类器置信度阈值合适吗？

答：随机抽取的 20 个例子暂无发现有语言识别错误，有 11 个文档是英语。试过分类器默认的置信度阈值合适。

四、mask_pii

1. 编写一个函数来屏蔽电子邮件地址，将所有电子邮件地址替换为 "|||EMAIL_ADDRESS|||"。

代码块

```
def run_mask_emails(text: str) -> tuple[str, int]:
    mask_email_text, cnt = re.subn(r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}', "|||EMAIL_ADDRESS|||", text)
    return mask_email_text, cntdef run_mask_emails(text: str) -> tuple[str, int]:
    mask_email_text, cnt = re.subn(r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}', "|||EMAIL_ADDRESS|||", text)
    return mask_email_text, cnt
```

2. 编写一个函数来屏蔽电话号码，将所有电话号码替换为 "|||PHONE_NUMBER|||"。

代码块

```
def run_mask_phone_numbers(text: str) -> tuple[str, int]:
    PHONE_REGEXES = {
        "zh": r"(\+?86)?1[3-9]\d{9}",                                  # 中国大陆手机号
        "en": r"(?<!\d)(\s*)((\+?1\s*[-.\s]*)?(\(\d{2,4}\)|\d{2,4})[-\s.]?\d{3,4}[-\s.]?\d{4})(?!\d)",  # 美国/加拿大
        "be": r"(?<!\d)(\s*)((\+?32)?\s*\(?0?\)?\s*(\d{1,2}[-\s.]?\d{2,3}[-\s.]?\d{2,3}[-\s.]?\d{2,3}))(?!\d)",
        "ja": r"(\+?81)?0\d{1,4}[\s-]?\d{1,4}[\s-]?\d{4}",          # 日本
        "ko": r"(\+?82)?0[1-9]\d{1,3}[\s-]?\d{3,4}[\s-]?\d{4}",  # 韩国
        "fr": r"(\+?33)?\s?[67]\d{8}",                                 # 法国
        "de": r"(\+?49)?1[5-7]\d{8}",                                  # 德国
        "ru": r"(\+?7)?9\d{9}",                                        # 俄罗斯
        "it": r"(\+?39)?3\d{9}",                                       # 意大利
        "id": r"(\+?62)?8\d{8,11}",                                    # 印尼
        "tr": r"(\+?90)?5\d{9}",                                       # 土耳其
        "th": r"(\+?66)?[689]\d{8}",                                   # 泰国
        "vi": r"(\+?84)?(3|5|7|8|9)\d{8}",                             # 越南
```

```
        "pt": r"(\+?351)?9[1236]\d{7}",                    # 葡萄牙
        "es": r"(\+?34)?6\d{8}",                           # 西班牙
        "uk": r"(\+?380)?
(39|50|63|66|67|68|73|91|92|93|94|95|96|97|98|99)\d{7}",  # 乌克兰
        "pl": r"(\+?48)?[5-8]\d{8}",                       # 波兰
        "ro": r"(\+?40)?7[2-8]\d{7}",                      # 罗马尼亚
        "bn": r"(\+?880)?1[3-9]\d{8}",                     # 孟加拉
        "hi": r"(\+?91)?[6789]\d{9}",                      # 印地语（印度）
        "ta": r"(\+?91)?[6789]\d{9}",                      # 泰米尔语（印
度）
        "te": r"(\+?91)?[6789]\d{9}",                      # 泰卢固语（印
度）
        "gu": r"(\+?91)?[6789]\d{9}",                      # 古吉拉特语（印
度）
        "ur": r"(\+?92)?3\d{9}",                           # 乌尔都语（巴基
斯坦）
        "fa": r"(\+?98)?9\d{9}",                           # 波斯语（伊朗）
        "he": r"(\+?972)?5\d{8}",                          # 希伯来语（以色
列）
        "az": r"(\+?994)?(50|51|70|77)\d{7}",              # 阿塞拜疆
        "ms": r"(\+?60)?1[0-9]\d{7,8}",                    # 马来语（马来西
亚）
        "nl": r"(\+?31)?6\d{8}",                           # 荷兰语
        "sv": r"(\+?46)?7[0236]\d{7}",                     # 瑞典语（瑞典）
    }
    total_cnt = 0
    new_text = text
    for regexe in PHONE_REGEXES.values():
        new_text, cnt = re.subn(regexe, r"\1|||PHONE_NUMBER|||", new_text)
        total_cnt += cnt
    return new_text, total_cnt
```

3. 编写一个函数来屏蔽 IPv4 地址，将所有 IP 地址替换为 "|||IP_ADDRESS|||"。

代码块

```
def run_mask_ips(text: str) -> tuple[str, int]:
    mask_ip_text, cnt = re.subn(r"\b(\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.(\d|
[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.(\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.(\d|[1-
9]\d|1\d{2}|2[0-4]\d|25[0-5])\b", "|||IP_ADDRESS|||", text)
    return mask_ip_text, cnt
```

4. 当这些过滤器被简单地应用于训练集时，你认为语言模型的下游可能会出现哪些问题？你如何缓解这些问题？

答：邮箱、手机号、IP地址敏感信息过滤器将其简单粗暴地替换为各自统一的 token 会对语言模型的下游产生下面的问题：（1）模型无法学到不同邮箱在语义、上下文中的区别；（2）模型失去生成真实邮箱结构的能力；（3）模型可能对这些 token 过拟合；（4）在推理/部署、问答、对话系统中，真实文本会包含真实的邮箱、手机号、IP地址，模型无法识别。缓解方法：（1）用结构化 token 替换，保留一些信息；（2）训练时以一定概率将敏感信息保留为原始字符串。

5. 对从 WARC 文件中提取的文本（通过之前实现的文本提取函数）运行 PII 屏蔽函数。查看 20 个随机替换示例；并给出一些误报和漏报的例子。

答：有一些电话号码没有被替换，例如：Ask an Expert: 1800 69 2225 1800 69 2225，Free Call: 1800 69 2225，(03) 9999 7735

五、harmful_content

1. 编写函数实现检测 NSFW 内容。

代码块

```
def run_classify_nsfw(text: str) -> tuple[Any, float]:
    text = text.replace('\n', '')
    model = fasttext.load_model(str(MODEL_PATH /
"jigsaw_fasttext_bigrams_nsfw_final.bin"))
    flag, p = model.predict(text, k=1)
    return str(flag[0]).split('__label__')[1], p[0]
```

2. 编写函数检测恶意言论。

代码块

```
def run_classify_toxic_speech(text: str) -> tuple[Any, float]:
    text = text.replace('\n', '')
    model = fasttext.load_model(str(MODEL_PATH /
"jigsaw_fasttext_bigrams_hatespeech_final.bin"))
    flag, p = model.predict(text, k=1)
    return str(flag[0]).split('__label__')[1], p[0
```

六、gopher_quality_filters

（a）根据下列要求，实现一个 Gopher 质量过滤器。

- 包含少于 50 或多于 100 个单词；
- 单词平均长度不在 3~10 的范围；
- 超过 30% 的行以 "..." 结尾；

- 包含至少一个字母的单词少于 80%。

代码块

```python
def run_gopher_quality_filter(text: str) -> bool:
    flag = True
    # 按空格切分文本
    words = text.split()
    new_words = []
    # 排除 words 里面非字母的字符
    for word in words:
        new_word = re.sub(r'[^a-zA-Z]', '', word)
        new_words.append(new_word)
    new_words = list(filter(lambda x:x != '', new_words))
    # 统计单词的数量
    word_cnt = len(new_words)
    # 如果单词数量小于 50 或者 大于 100,000，则文本不通过 Gopher 质量过滤器
    if word_cnt < 50 or word_cnt > 100000:
        flag = False
    # 如果单词数量占比少于 80%，则文本不通过 Gopher 质量过滤器
    if len(words) == 0:
        flag = False
    if len(words) > 0 and len(new_words) / len(words) < 0.8:
        flag = False
    # 计算单词的平均长度
    words_len = 0
    for it in new_words:
        words_len += len(it)
    avg_words_len = words_len / len(new_words)
    # 如果单词平均长度小于 3 或者 大于 10，则文本不通过 Gopher 质量过滤器
    if avg_words_len < 3 or avg_words_len > 10:
        flag = False
    lines = text.splitlines()
    if lines:
        end_with_dots = sum(1 for line in lines if
line.strip().endswith("..."))
        # 如果超过 30% 的行以 ... 结尾，则文本不通过 Gopher 质量过滤器
        if end_with_dots / len(lines) > 0.3:
            flag = False
    return flag
```

（b）对从 WARC 文件中提取的文本（通过之前实现的文本提取功能）运行基于规则的质量过滤器。查看 20 个随机示例，并将过滤器的预测与你自己的判断进行比较。评论任何质量过滤器与你的判断不同的情况。

答：有一些网站的首页主要是介绍销售商品内容的，例如下面这个例子。我认为这些不应该通过质量过滤器，但是它们通过了。

代码块

```
    • About Will Garcia
    • WORK WITH ME!
    • Hotels
    • Restaurants
    • Travel
    • Events
    • Fashion
    • Advertise With Us
Will Explore Philippines and World
Exploring the Best of the Philippines and the World
    • Restaurants
    • Travel and Leisure

Indulge with Half Dozen of J. Co Donuts for only 1 Peso via GrabFood

Posted on June 16, 2021June 16, 2021by willgarcia

You deserve a sweet treat! Get half dozen donuts for only 1 Peso from J.Co
with a minimum order of a dozen donuts when you order via GrabFood.

I personally love Alcapone, Avocado Di Caprio and Green Tease. I love J.Co for
its one of the premium donut brands that make delicious sweet treats that
satisfy my cravings.

Hurry! Grab your phones and order at GrabFood. Enjoy this amazing deal this
June 16-17, 2021, 2pm to 6pm! *Available in selected stores only.

Half dozen pre-assorted flavors: Alcapone, Oreology, Rainbow Strawberry, Coco
Loco, Green Tease and Joccino.

*Please see list of participating JCO stores:

•Big Ben Lipa (Batangas)
•SM Baguio
•Ayala Technohub (Baguio)
•SM Pampanga
•Marquee Mall (Pampanga)
•SM Olongapo
•SM City Cebu
•Ayala Center Cebu
•Calyx Center Cebu IT Park
•Robinsons Iloilo
```

```
•SM City Iloilo
•SM City Davao
•Abreeza Mall (Davao)
•Centrio Mall (Cagayan De Oro)
•KCC Mall de Zamboanga
•Veranza Mall (General Santos)
•Ayala Malls Legazpi

Share this:

  • Facebook
  • Twitter
  • Tumblr
  • Pinterest
  • LinkedIn
  • Email

Like this:

Like Loading...

Related

No comments yet

Leave a Reply Cancel reply

Post navigation

Satisfy your cravings with 1 Peso Delivery Fee on GrabFood Signatures
Crime-buster and corruption fighter

  • Advertise and Collaborate With Us

Activello Theme by Colorlib Powered by WordPress
Loading Comments...
    %d
```

七、quality_classifier

（a）训练一个质量分类器，给定文本，返回对应的质量分数。我的思路如下：

- 先从英语维基百科页面上的 4350 万个外部链接列表随机抽取 2 万个 url。
- 使用 wget 命令以 WARC 格式抓取 url 里面的内容。

- 从 WARC 文件中提取的文本（通过之前实现的文本提取功能）后，使用之前实现的语言识别功能过滤非英语的文本。
- 通过正则表达式过滤一些包含 403，404，被封爬虫提示的文本。
- 过滤非字母占比高于平均值的文本，确保最后所有文本长度大于 200。
- 将上面得到的文本作为 wiki 样本，common crwal 的文本作为 cc 样本，构建数据集，使用 fasttext 训练分类器。

（b）编写一个函数，将页面标记为高质量或低质量，并在标签中提供置信度分数。

代码块

```
def run_classify_quality(text: str) -> tuple[Any, float]:
    text = text.replace('\n', '')
    model = fasttext.load_model(str(MODEL_PATH / "quality_classifier.bin"))
    flag, p = model.predict(text, k=1)
    return str(flag[0]).split('__label__')[1], p[0]
```

八、exact_deduplication

要求：编写一个函数，接收一个文件路径列表，并对其进行精确的行重复数据删除。该函数首先计算语料库中每行的频率，并使用哈希值来减少内存占用，然后重写每个文件，仅保留其唯一行。

注意：

1. 这里是把所有文件看成一个大集合，找出大集合里重复出现过的行，把所有出现超过一次的行从每个文件中都删掉；
2. 如果输出的文件是空的话，那就是空输出，如果输出文件非空，那么要在末尾保留换行符。

代码块

```
def run_exact_line_deduplication(
    input_files: list[os.PathLike], output_directory: os.PathLike
):
    line_cnt = {}
    # 第一次遍历，将目录下所有文件的行用 md5 做哈希映射并统计次数
    for input_file in input_files:
        with open(input_file, "r", encoding="utf-8") as f:
            file_lines = f.read().splitlines()
        for line in file_lines:
            line_hash = hashlib.md5(line.encode()).hexdigest()
            line_cnt[line_hash] = line_cnt.get(line_hash, 0) + 1
    output_directory.mkdir(parents=True, exist_ok=True)
    # 第二次遍历，对每一行查看它的 md5 哈希值对应的计数是否大于 1
    for input_file in input_files:
        output_file = []
```

```
        with open(input_file, "r", encoding="utf-8") as f:
            file_lines = f.read().splitlines()
        for line in file_lines:
            line_hash = hashlib.md5(line.encode()).hexdigest()
            if (line_cnt[line_hash] > 1):
                continue
            else:
                output_file.append(line)
        content = "\n".join(output_file).rstrip("\n")
        with open(str(output_directory / Path(input_file).name), "w",
encoding="utf-8") as f:
            if content:
                f.write(content + "\n")
            else:
                f.write("")
```

九、minhash_deduplication

要求：编写一个函数，接受一个文件路径列表，并使用最小哈希函数 (minhash) 和局部哈希函数 (LSH) 执行模糊文档去重。具体来说，你的函数应该为给定路径列表中的每个文档计算最小哈希签名，使用给定波段数的局部哈希函数识别候选重复项，然后计算候选重复项之间的真实 ngram Jaccard 相似度，并删除超过给定阈值的重复项。为了提高召回率，在计算最小哈希签名和/或比较 Jaccard 相似度之前，应先对文本进行规范化，包括小写化、删除标点符号、规范化空格、删除重音符号，以及应用 NFD Unicode 规范化。

整个过程比较复杂，可以概括为以下几个主要步骤：

1. 文本规范化

文本规范化操作顺序如下：

（1）应用 NFD Unicode 规范化将带有重音符号的预组合字符分解成基本字母和独立的组合字符（例如：'ñ' 分解为 'n' 和 '~'），遍历 NFD 后的字符串过滤组合字符（例如 '´', '~', '¨' 等）；

（2）将所有英文字母转为小写；

（3）将标点符号替换成单个空格；

（4）将多个连续的空格替换成 1 个空格。

代码块

```
# applying NFD unicode normalization and removing accents
nfd_text = unicodedata.normalize('NFD', text)
no_accents_text = ''.join([char for char in nfd_text if
unicodedata.category(char) != 'Mn'])
# lowercasing
text_lower = no_accents_text.lower()
```

```
# removing punctuation
text_nopun = re.sub(r'[^\w\s]', ' ', text_lower, flags=re.UNICODE)
# normalizing whitespaces
text_nospace = re.sub(r'\s+', ' ', text_nopun).strip()
```

2. 将规范化后的文档拆分成连续的词 N-gram 集合

代码块

```
def get_ngrams(text, ngrams):
    n_grams = set([])
    for i in range(len(text) - ngrams + 1):
        n_grams.add(text[i : i+ngrams])
    return n_grams

# 将文件分解成 n-grams 集合 S
S = get_ngrams(text_nospace, ngrams)
```

3. 生成 minihash signatures

操作如下：

（1）生成 num_hashes 个独立于 N-gram 集合预先定义的哈希函数，具体是用 h_it = abs(hash(it + random_string[i]))，it 是一个 N-gram 集合里面的其中一个元素。在全局定义一个 num_hashes 大小的 random_string 就能保证所有文件的 N-gram 都使用相同的哈希函数；

（2）构建一个长度为 num_hashes 的 signature 表，初始值设置为很大的数；

（3）对于 signature 表的每一个元素 i 以及 N-gram 集合 S 的每一个元素 it，使用 h_it = abs(hash(it + random_string[i])) 计算对应的哈希值。然后用 signature[i] = min(signature[i], h_it) 更新 signature。完成一个文件对应的 N-gram 集合 S 遍历后，就得到该文档 num_hashes 维的 minihash signature。

代码块

```
random_strings = [str(random.random()) for _ in range(num_hashes)]
def get_minhash_sign(S, num_hashes, random_strings):
    signature = []
    for i in range(num_hashes):
        mini_hash = sys.maxsize
        for it in S:
            h_it = abs(hash(it + random_strings[i]))
            if h_it < mini_hash:
                mini_hash = h_it
        signature.append(mini_hash)
    return signature
```

4. 应用 LSH 识别可能相似的文档候选对

操作如下：

（1）将每个文档的 num_hashes 维 signature 分割成 num_bands 个 bands；

（2）对每个 bands 使用 sha1 哈希函数映射到一个桶；

（3）遍历所有的桶，如果它包含的文档数量大于 1，则提取放入新的集合 dop_doc；

代码块

```
r = int(num_hashes / num_bands)
band_bucket = {i: {} for i in range(num_bands)}
for file_name, signature in file_sign.items():
    for num_band in range(num_bands):
        left_idx = num_band * r
        right_idx = left_idx + r
        sign_vector = tuple(signature[left_idx : right_idx])
        sign_hash = hashlib.sha1(str(sign_vector).encode("utf-8")).hexdigest()
        if sign_hash not in band_bucket[num_band]:
            band_bucket[num_band][sign_hash] = []
        band_bucket[num_band][sign_hash].append(file_name)

dup_doc = set()
for num_band in range(num_bands):
    for sign_hash, file_list in band_bucket[num_band].items():
        if len(file_list) > 1:
            for i in range(len(file_list)):
                for j in range(i + 1, len(file_list)):
                    pair = tuple(sorted((file_list[i], file_list[j])))
                    dup_doc.add(pair)
```

5. 计算 Jaccard 相似度并进行文档聚类

操作如下：

（1）使用输入文档列表初始化并查集 uf；

（2）遍历 dop_doc 集合里面的候选文档对，取出它们的 minihash signature，计算 Jaccard 相似度，如果相似度大于预设的阈值，则对该候选对执行 union 操作。

代码块

```
# 使用所有文档的名称初始化并查集
uf = UnionFind(list(file_sign.keys()))

for pair in dup_doc:
    file_sign1 = file_sign[pair[0]]
    file_sign2 = file_sign[pair[1]]
```

```
        jaccard_sim = (sum(1 for x, y in zip(file_sign1, file_sign2) if x == y)) / len(file_sign1)
        # 如果两个文档的 Jaccard 相似度超过设定阈值
        if jaccard_sim > jaccard_threshold:
            # 则将这两个文档 union 成一个更大的簇
            uf.union(pair[0], pair[1])
```

6. 遍历 uf.groups() 的所有 keys，读取这些 keys 对应的文档，然后写入 output_directory 目录

代码块

```
output_directory.mkdir(parents=True, exist_ok=True)
# 遍历所有簇，因为 union 操作后，所有相似的文档都合并到一个簇，所有只需将所有的 keys 写入新的目录即可

for k in uf.groups().keys():
    in_path = Path(k)
    out_path = str(output_directory / in_path.name)
    with open(in_path, "r", encoding="utf-8") as f_in:
        text = f_in.read()
    with open(out_path, "w", encoding="utf-8") as f_out:
        f_out.write(text)
```