

HW #01: Inverted Index Lib

1. Описание задания	2
1.1. Инвертированный индекс (Inverted Index)	2
1.2. Входные данные	3
1.3. Требования к реализации	4
2. Рекомендации	5
3. Критерии оценивания	5
4. Инструкция по отправке задания	6
5. FAQ (часто задаваемые вопросы)	8
6. Дополнительные задания (не на оценку)	9

1. Описание задания

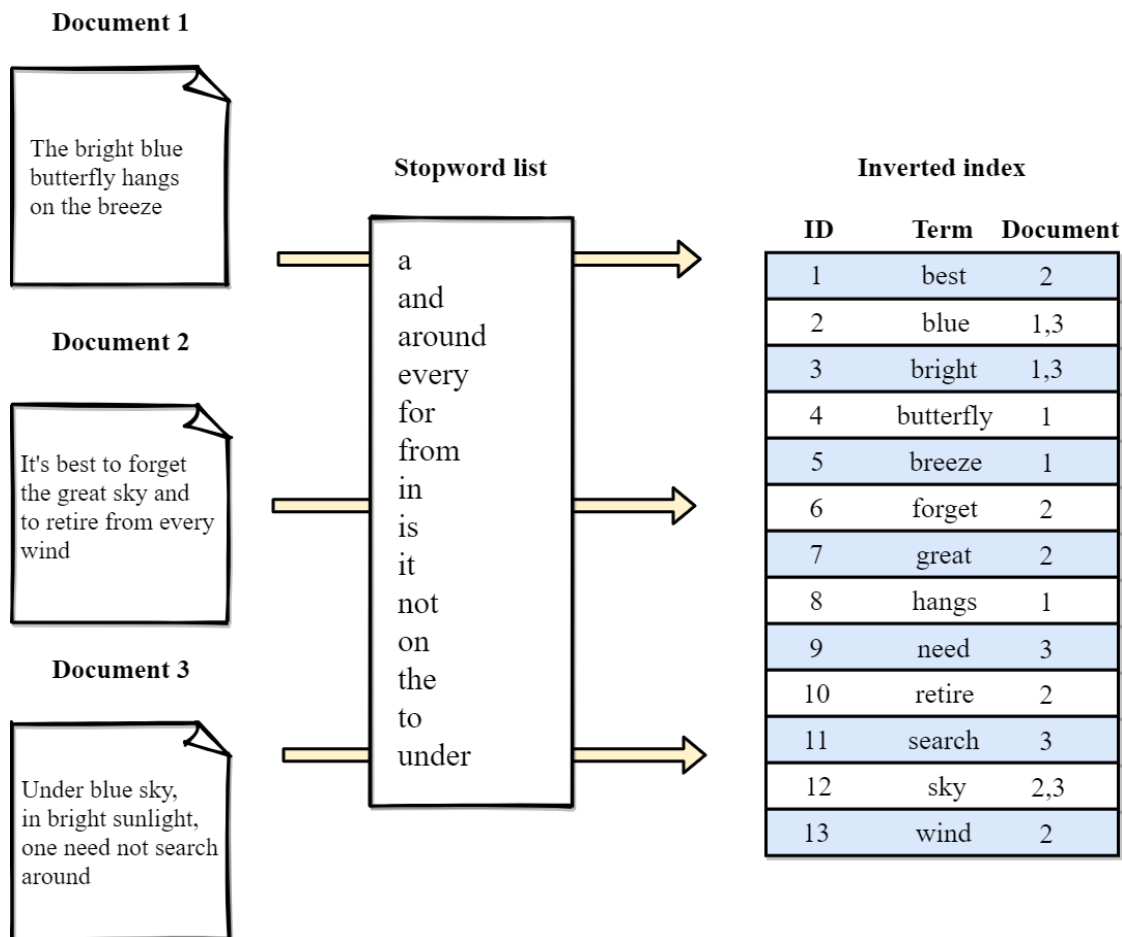
В этом задании вам нужно написать библиотеку по построению инвертированного индекса. Цель задания - приучить себя сначала писать тесты, а уже потом реализацию (TDD).

Правила:

- TDD - сначала тесты, потом реализация;
- Рефакторинг не смешиваем с добавлением функциональности;
- Fuzz/Stress Testing для валидации эффективной реализации.

1.1. Инвертированный индекс (Inverted Index)

Инвертированный индекс представляет собой словарь, где ключами являются слова (термы), а значениями - списки идентификаторов документов, в которых указанный терм встречается (см. Рис. 1).



(Рис. 1) Инвертированный индекс

Такая структура позволяет поисковым системам найти страницы в интернете, которые могут быть релевантны пользовательскому запросу. Вам будет предоставлен датасет из документов и по этому датасету нужно построить инвертированный индекс. Библиотека должна предоставить возможность:

1. Считать датасет в оперативную память;
2. Разбить каждый документ на термы (слова);
3. ~~Удалить все стоп-слова;~~¹
4. Построить инвертированный индекс;
5. Сохранить инвертированный индекс на диск;
6. Загрузить инвертированный индекс с диска;
7. Найти документы, соответствующие заданному поисковому запросу (если в запросе указаны слова "Python" и "code", то нужно вывести только те документы, которые содержат **оба** этих слова).

На Рис. 1 указаны стоп-слова - это слова, которые встречаются практически в каждом документе. В связи с этим, хранение списка документов, в которых встречается это стоп-слово, не только практически бессмысленно, но еще и болезненно, поскольку хранение этой информации занимает много места в оперативной памяти (или на жестком диске). Бывают и исключения, иногда нужно находить документы на запросы, которые состоят полностью из стоп-слов, например "to be or not to be". Но решение этой проблемы выходит за рамки нашего курса. Короткий ответ - можно хранить вместо термов биграммы из стоп-слов (например "to be", "or not" и т.п.) и искать по ним. Длинный ответ - почитайте релевантную литературу по Information Retrieval (информационному поиску).

Любимая книга автора курса на эту тему:

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.
- Доступна для бесплатного скачивания: <https://nlp.stanford.edu/IR-book/>

1.2. Входные данные

Дамп Википедии

- Формат: текст
- В каждой строке находятся следующие поля, разделенные знаком табуляции:

¹ В этом задании стоп-слова пока не учитываем, считаем их обычными словами (термами).



1. INT - id статьи,
2. STRING - текст статьи,

Пример:

```
12  Anarchism           Anarchism is often defined as a political
philosophy which holds the state to be undesirable, unnecessary, or
harmful.
```

1.3. Требования к реализации

Возьмите за основу Starter реализации библиотеки:

- [github:big-data-team/python-course/.../inverted_index_starter.py](https://github.com/big-data-team/python-course/blob/master/inverted_index_starter.py)

Вам нужно реализовать объявленные функции в Starter'e. Поскольку инвертированный индекс по факту представляет собой словарь, то его хранение на жестком диске рекомендуется выполнить с помощью стандартной библиотеки json (см. `json.load(s)` и `json.dump(s)`).

Добавьте пустой `test_inverted_index.py` и добейтесь уровня покрытия тестами 80% (см. `pytest --cov` и `--cov-branch`). При разработке пользуемся идеологией TDD. Примеры тестов:

1. Можно построить инвертированный индекс на основе данных в runtime Python;
2. Можно построить инвертированный индекс на основе данных в файловой системе (используем заготовленный файл);
3. Инвертированный индекс можно выгрузить в файловую систему и из нее загрузиться (используем `tmpdir` в тестах);

Для гарантии одинаковых результатов, необходимо использовать следующую конструкцию для токенизации статьи на слова:

```
doc_id, content = line.lower().split("\t", 1)
doc_id = int(doc_id)
words = re.split(r"\W+", content)
```

Ожидаемая сигнатура функций:

- `InvertedIndex.query(self, words: List[str]) -> List[int]`
- `InvertedIndex.dump(self, filepath: str) -> None`
- `InvertedIndex.load(cls, filepath: str) -> InvertedIndex`



- `load_documents(filepath: str) -> Dict[int, str]`
индексы документов кастуем к `int`'ам
content не парсим на слова, только строку достаем и отрезаем `"\n"` в конце строк
- `build_inverted_index(load_documents(filepath: str)) -> InvertedIndex`

После реализации тестов и функционала библиотеки убедитесь, что ваш код является "чистым" (Clean Code):

- изучите [PEP-8](#);
- проверьте качество вашего кода до и после исправления с помощью `pylint`.

2. Рекомендации

Рекомендуем настроить виртуальное окружение Python с нужными версиями библиотек:

1. Если еще не установлено, то установите `conda`
<https://docs.conda.io/projects/conda/en/latest/user-guide/install/>
2. Настройте окружение для разработки на основе `README.md` курса
<https://github.com/big-data-team/python-course>
3. Скачайте необходимые датасеты для выполнения задания
<https://github.com/big-data-team/python-course#study-datasets>

3. Критерии оценивания

Балл за задачу складывается из:

- **20%** - правильная реализация поиска
- **30%** - возможность выгрузки и загрузки инвертированного индекса на/с жесткого диска
 - запросы типа `build` и `query` должны обрабатываться в течение 5 минут на представленном в описании ДЗ датасете
- **40%** - качество покрытия тестами
 - оценка качества проводится автоматически вызовом `pytest`:
 - `PYTHONPATH=. pytest -v --cov=task_*_inverted_index_lib test_*_inverted_index_lib.py`
 - уровень покрытия тестами должен быть выше 80%
 - проверяем код Python версии 3.7 с помощью `pytest==6.0.1`
 - точная формула: $40\% \times \min([\text{test_coverage} / 0.8], 1.0)$
- **10%** - поддерживаемость и читаемость кода

- в общем случае см. Clean Code и [Google Python Style Guide](#)
- оценка качества будет проводиться автоматическим вызовом pylint:
 - `pylint task_*.py`
 - качество кода должно оцениваться выше 8.0 / 10.0
 - проверяем код Python версии 3.7 с помощью `pylint==2.5.3`
 - точная формула: $10\% \times \min([\text{lint_quality} / 8.0], 1.0)$

Discounts (скидки и другие акции):

- **100%** за плагиат в решениях (всем участникам процесса)
- **100%** за посылку решения после hard deadline
- **30%** за посылку решения в после soft deadline и до hard deadline
- **5%** за каждую посылку после 2й посылки в день (каждый день можно делать до 2х посылок без штрафа)

лучший балл с 1-й попытки: 100%

лучший балл со 2-й попытки: 100%

лучший балл с 3-й попытки: 95%

лучший балл с 4-й попытки: 90%

Для подсчета финальной оценки **всегда** берется **последняя** оценка из Grader.

4. Инструкция по отправке задания

Оформление задания:

- Код задания (Short name): **HW01:InvertedIndex Lib**
- Выполненное ДЗ запакуйте в архив `PY-MADE-2021-Q4_<Surname>_<Name>_HW#.zip`, пример -- `PY-MADE-2021-Q4_Dral_Alexey_HW01.zip`. (Проверяйте отсутствие пробелов и невидимых символов после копирования имени отсюда.²) Если ваше решение лежит в папке `my_solution_folder`, то для создания архива `hw.zip` на Linux и Mac OS выполните команду³:
 - `zip -r hw.zip my_solution_folder/*`
- На Windows 7/8/10: необходимо выделить все содержимое директории `my_solution_folder/` нажать правую кнопку мыши на одном из выделенных объектов, выбрать в открывшемся меню "Отправить >", затем "Сжатая ZIP-папка". Теперь можно переименовать архив.
- Решение задания должно содержаться в одной папке.
- Перед проверкой убедитесь, что дерево вашего архива выглядит так:

² Онлайн инструмент для проверки: <https://www.soscisurvey.de/tools/view-chars.php>

³ Флаг -r значит, что будет совершен рекурсивный обход по структуре директории



- | PY-MADE-2021-Q4_<Surname>_<Name>_HW01.zip
- | ---- task_<Surname>_<Name>_inverted_index_lib.py
- | ---- test_<Surname>_<Name>_inverted_index_lib.py
- | ---- *.txt⁴
- При несовпадении дерева вашего архива с представленным деревом, ваше решение не будет возможным автоматически проверить, а значит, и оценить его.
- Для того, чтобы сдать задание необходимо:
 - Зарегистрироваться и залогиниться в сервисе [Everest](#)
 - Перейти на страницу приложения: [MADE Python Grader](#)
 - Выбрать вкладку Submit Job (если отображается иная).
 - Выбрать в качестве "Task" значение: **HW01:InvertedIndex Lib**⁵
 - Загрузить в качестве "Task solution" файл с решением
 - В качестве Access Token указать тот, который был выслан по почте, если письмо не пришло, пишите в Telegram @bdt_support
- **Перед отправкой задания**, оставьте, пожалуйста, отзыв о домашнем задании по ссылке: https://rebrand.ly/pymade2021q4_feedback_hw. Это позволит нам скорректировать учебную нагрузку по следующим заданиям (в зависимости от того, сколько часов уходит на решение ДЗ), а также ответить на интересующие вопросы.

Внимание: если до дедлайна остается меньше суток, и вы знаете (сами проверили или коллеги сообщили), что сдача решений сломана, обязательно сдайте свое решение, прислав нам ссылку на выполненное задание (Job) на почту с темой письма "Short name. ФИО.". Например: "HW01:InvertedIndex Lib. Иванов Иван Иванович." Таким образом, мы сможем увидеть какое решение у вас было до дедлайна и сможем его оценить. Пример ссылки:

- <https://everest.distcomp.org/jobs/67893456230000abc0123def>

Любые вопросы / комментарии / предложения пишите согласно [предложениям](#) на портале.

Всем удачи!

⁴ Архив с тестовыми данными должен занимать **менее 1МБ** пространства на жестком диске

⁵ Сервисный ID: python.inverted_index_lib

5. FAQ (часто задаваемые вопросы)

"You are not allowed to run this application", что делать?

Если Вы видите надпись "You are not allowed to run this application" во вкладке Submit Job в Everest, то на данный момент сдача закрыта (нет доступных для сдачи домашних заданий, по техническим причинам или другое). Попробуйте, пожалуйста, еще раз через некоторое время. Если Вы еще ни разу не сдавали, у коллег сдача работает, но Вы видите такое сообщение, сообщите нам об этом.

Grader показывает 0 или < 0 , а отчет (Grading report) не помогает решить проблему

Ситуации:

- система оценивания показывает оценку (Grade) < 0 , а отчет (Grading report) не помогает решить проблему. Пример: в случае неправильно указанного access token система вернет -401 и информацию о том, что его нужно поправить;
- система показывает 0 и в отчете (Grading report) не указано, какие тесты не пройдены. Пример: вы отправили невалидный архив (rar вместо zip), не приложили нужные файлы (или наоборот приложили лишние - временные файлы от Mac OS и т.п.), рекомендуется проверить содержимое архива в консоли:

```
unzip -l your_solution.zip
```

Если Вы столкнулись с какой-то из них присылайте ссылку на выполненное задание (Job) в чат курса. Пример ссылки:

<https://everest.distcomp.org/jobs/67893456230000abc0123def>



6. Дополнительные задания (не на оценку)

Расширим функционал библиотеки и для этого предварительно проведем рефакторинг кода:

1. Предоставьте возможность расширять функционал библиотеки инвертированного индекса с помощью Стратегии. Добавьте базовый класс стратегии реализации load/dump инвертированного индекса:

- а. [github:big-data-team/python-course/.../storage_policy.py](https://github.com/big-data-team/python-course/blob/master/storage_policy.py)

Создайте класс JsonStoragePolicy и перенести необходимую функциональность туда. Убедитесь, что все тесты проходят. Следите за качеством кода с помощью pylint (см. <https://github.com/big-data-team/python-course#howtos>).

2. Реализуйте дополнительные стратегии хранения индекса с помощью библиотек pickle и zlib. Напишите тесты для проверки валидности реализации на основе JsonStoragePolicy.
3. Поделитесь в канале группы, какого сжатия удалось добиться с помощью выбранных библиотек по сравнению с json. Для удобства поиска можно использовать хеш-теги: #inverted_index #compression_challenge