# Файловая структура

У проекта tve есть определенная файловая структура:

Базовая директория содержит 2 подпапки, файлы со скриптами, а также jupyter ноутбуки для быстрого прототипирования или наглядной оценки качества, а также служебные файлы.  
Подпапки:

* tve – папка со всей логикой пакета
* data – стандартная папка с входными и выходными данными экспериментов, заранее заготовленными промптами, файлом с стоп словами, а также папкой с логами отработки программ

## Jupyter notebooks:

* validate.ipynb – ноутбук для визуализации экспериментов разных систем. Читает формат файлов выводов экспериментов
* vectorization.ipynb – ноутбук для работы с энкодерными моделями и векторными базами данных с наглядной визуализацией и созданием таблиц с метриками

Скрипты:

* get\_keywords.py – скрипт для получения ключевых слов при помощи YAKE, запускаемых локально LLM по методу KeyBERT. Для выбора того, какая система будет доставать ключевые слова нужно раскомментировать соответствующие элементы или добавить свои инстансы базового класса KeyWordExtractorBase (например YAKExtractor или KeyBERTExtractor)

в спискок *extractors* на 59 строке

Аргументы для запуска:

-i/--input <папка> - название папки содержащей документы или папки с одним документом. ВНИМАНИЕ: папка расположена по пути “[BASE\_PATH/DOCS\_PATH](#_Переменные_проекта)/<папка> ”. Обязательный параметр

-o/--output <папка> - название папки куда сохранять результаты получения ключевых слов. Необязательный параметр, если не указан то папка будет называться также как и *input.* ВНИМАНИЕ: папка расположена по пути “[BASE\_PATH/DATA\_PATH](#_Переменные_проекта)/<папка> ”.

-n/--number <число> - ограничение числа документов на данный запуск скрипта

-w/--num-of-workers <число> - количество асинхронно работающих задач (полезно когда используется доступ к API) стандартно: 1

--no-rewrite – необязательный параметр, который указывает, что нужно очищать старые результаты экспериментов (расположены в папке output)

--no-skip – необязательный параметр, который указывает, что НЕ нужно пропускать обработку патентного документа, если его результаты для данной системы уже имеются в папке *output*. (Полезно, когда обработка датасета производится в несколько последовательных запусков скриптов, сначала для одной системы затем для других)

* get\_keywords\_llm.py – скрипт который позволяет получать ключевые слова при помощи API больших языковых моделей, а именно YandexGPT, GigaChat, ChatGPT. Позволяет получать базовый термвектор или сразу расширенный тервектор (за один промпт)

Аргументы для запуска:

-i/--input <папка> - см. Описание для get\_keywords.py

-o/--output <папка> - см. Описание для get\_keywords.py

-n/--number <число> - см. Описание для get\_keywords.py

-w/--num-of-workers <число> - см. Описание для get\_keywords.py

--no-rewrite – см. Описание для get\_keywords.py

--no-skip - см. Описание для get\_keywords.py

-m/--model <буква обозначающая модель> - выбор модели (y – YandexGPT, g – GigaChat, c - ChatGPT)

-p/--prompt <имя> - выбор используемого промпта, хранящегося в папке “[BASE\_PATH/PROMPTS\_PATH](#_Переменные_проекта)” в формате json.

-j – опциональный флаг для определения, является ли ответ модели json объектом. Стандартно, ожидается что модель выдает ключевые слова, разделенные запятой и пробелом “, ”.

* get\_translations.py – скрипт который позволяет использовать проприетарные LLM (GigaChat, YandexGPT, ChatGPT) для расширения/перевода базовых термвекторов.

Аргументы для запуска:

-m/--models <строка букв обозначающих модели> (y – YandexGPT, g – GigaChat, c – ChatGPT, P - Promt) какие модели будут исполнять запросы. Обязательный параметр

-i/--input <папка> - название папки содержащей результаты работы остальных скриптов в JSON. ВНИМАНИЕ: папка расположена по пути “[BASE\_PATH/DATA\_PATH](#_Переменные_проекта)/<папка> ”. Обязательный параметр

-o/--output <папка> - см. Описание для get\_keywords.py. (стандартно: input + ”\_trans”)

-n/--number <число> - см. Описание для get\_keywords.py

-w/--num-of-workers <число> - см. Описание для get\_keywords.py

--no-rewrite – см. Описание для get\_keywords.py

--no-skip - см. Описание для get\_keywords.py

--sleep <время в секундах> - время паузы между совершением каждого запроса

* get\_relevant.py – скрипт который позволяет получать результаты поиска по термвекторам полученным в результате исполнения остальных скриптов. Выбор поисковой платформы производится в коде (стандартно выбран ElasticSearch, поэтому скрипт нужно совершать на локальной машине). ВНИМАНИЕ, за один запрос в систему не посылается более 200 терминов, стандартно выдаётся топ-50 самых релевантных документов.

Аргументы для запуска:

-o/--output <папка> - см. Описание для get\_keywords.py. (стандартно: input + ”\_rel”)

-n/--number <число> - см. Описание для get\_keywords.py

-w/--num-of-workers <число> - см. Описание для get\_keywords.py

--no-rewrite – см. Описание для get\_keywords.py

--no-skip - см. Описание для get\_keywords.py

-i/--input <папка> - название папки содержащей результаты работы остальных скриптов в JSON. ВНИМАНИЕ: папка расположена по пути “BASE\_PATH/DATA\_PATH/<папка> ”. Обязательный параметр

-t/--timeout <число> - время таймаута для одного запроса в секундах

--wo-kws – параметр, указывающий, требуется ли в файл результатов поиска записывать термвектора, по которым производился поиск (Полезно для уменьшения размера файлов)

--scores – параметр, указывающий, использовать ли веса терминов в поисковой платформе (если таковы имеются в файлах)

* format.py – скрипт для преобразования результат скрипта get\_relevant.py в формат для утилиты оценки качества.

Аргументы для запуска:

-i/--input <папка> – папка с работы get\_relevant.py скрипта. Обязательный параметр.

-o/--output <папка> – папка для вывода результата в формате подходящем для утилиты оценки качества. Обязательный параметр.

-p/--priority – опциональный флаг для определения, является ли документ скрипта приоритетной заявкой. В таком случае в качестве id берется не из самого документа а из второго документа кластера (result[“cluster”][1]). Запись нужного документа на второе место в кластере необходимо самостоятельно делать на этапе предварительной подготовки документов или позднее писать скрипт для обновления .json файлов с результатами.

# Переменные проекта

В проекте используются различные переменные, которые должны быть прописаны в локальном *.env* файле.

Переменные стандартизирующие пайплайн обработки данных:

* BASE\_PATH – Базовый путь данных, где находятся нижеперечисленные папки (стандартно “data”)
* DOCS\_PATH – Папка где находятся папки с JSON/XML документами (стандартно “data”)
* DATA\_PATH – Папка где находятся папки с результатами экспериментов (в формате json)
* PROMPTS\_PATH – Путь где находятся промты для Больших языковых моделей.
* LOGS\_PATH – Путь где находятся логи выполнения скриптов.

А также переменные содержащие информацию для некоторых классов:

* FIPS\_API\_KEY – API ключ для сервиса «Поисковая платформа»
* ES\_URL – URL сервера ElasticSearch
* CACHE\_DIR – Базовая папка для кэширования
* PROMT\_IP – IP сервера Промпт

# Пример использования:

Эксперимент по получению базовых ТВ при помощи YAKE, ChatGPT, YandexGPT, последующему их расширению при помощи PROMT, ChatGPT, YandexGPT, а также представлению данных в нужном виде для утилиты оценки качества.

python get\_keywords.py -i 2025\_docs -o 2025\_kws

python get\_keywords\_llm.py -i 2025\_docs -o 2025\_kws -m y -p ru\_kws -w 10

python get\_keywords\_llm.py -i 2025\_docs -o 2025\_kws -m c -p en\_kws -w 20

python get\_translations.py -I 2025\_kws -o 2025\_extended -m pyc

python get\_relevant.py -I 2025\_extended -o 2025\_rel --wo-kws -w 10 -t 120

python format.py -I 2025\_rel -o 2025\_res

# Описание существующих каталогов:

## Коллекции документов (docs):

* 1000\_priority… – эксперимент с переведенными американскими заявками, приоритетными для которых были российские
* …сlusters… - кластеры
* 80\_skolkovo\_ru – документы из сколково
* …ru – русские документы
* Embeddings – документы выбранные для экспериментов по построению эмбеддингов

## Результаты (data)

в том числе и промежуточные

* …kws – базовые термвектора
* …ext – расширенные/переведенные термвектора
* …rel – релевантная выдача
* …res – данные в формате для утилиты оценки качества
* …us… - исходный документ американский
* …ru… - исходный документ российский
* Во всех папках 1000\_priority\_us… кроме 1000\_priority\_us\_llm… использован метод получения сначала базовых термвектора на английском с последующим расширением при помощи LLM. В 1000\_priority\_ru… также используется ПРОМТ.
* В 1000\_priority\_us\_llm… использован метод получения сразу расширенного термвектора при помощи LLM
* Embeddings – результаты экспериментов по построению эмбеддингов