Запросы:

1. **Ракетный корпус** *(на илл.)* княжества Майсур в 1780 году нанёс британской армии одно из самых тяжёлых поражений.
2. От 500-килограммовой бомбы жилой дом **будущего академика** в годы войны спасла бочка с вином.
3. Согласно «Повести о житии Александра Невского», Бог — **не в силе, а в правде**.

Описание:

Содержательный текст из фактов и статей поместился в файлы queries.txt и docs.txt соответственно (каждый факт/запрос с новой строки). Разбиение статей на предложения производилось с помощью предложенной библиотеки razdel. Лемматизация с помощью библиотеки pymystem. Разбиения предложений на токены: токен непрерывная последовательность букв и цифр. Но текстовому корпусу вычислил значений idf лемм. Для предложений вычислил 2 варианта tf и получил окончательные векторные представления умножив на idf.

Запуск программы: python <doc file> <query file> <top k>

- <doc file> - файл с текстовым корпусом.
- <query file> - файл с фактами/запросами.
- <top k> - число наиболее подходящих под запрос предложений, которые будут указаны в отчете.

Результат: файлы report_m_1.txt, report_m_2.txt со списками подходящих предложений для двух вариантов.

В архиве помимо кода приложены файлы docs.txt и queries.txt, на которых выполнялся запуск. Файлы report_m_1.txt, report_m_2.txt для top_k = 10.

Выводы:

Для обоих вариантов tf и для всех запросов первые 5 предложение сортируются одинакового. Для второго запроса верно определяется только самое релевантное предложение, остальные выбираются из статьи, относящейся к первому запросу, из-за схожих тематик. Сложно выделить, какой вариант лучше, т.к. результаты очень похожи. Для 1 и 3 запросов находятся релевантные предложения, а для 2 находится только самое релевантное предложение, т.к. оно единственное подходит по запрос.