ТЕСТОВОЕ ЗАДАНИЕ СТАЖИРОВКИ JAVA-РАЗРАБОТЧИКОВ В RENUE

Классификатор текста

Постановка задачи

Требуется написать консольное Java-приложение (JDK 17), реализующее простой текстовый классификатор в чат-боте.

Приложение получает на вход при запуске CSV-файл со списком возможных отчетов, имеющих такие свойства, как идентификатор формата GUID, кодовое имя, описание.

Приложение получает текстовый запрос от пользователя в свободной форме. Например, «Предоставь мне отчет об уплаченных процентах по вкладу», по которому ищутся наиболее подходящие строки из файла по описанию, GUID-ы которых выводятся пользователю в ответе.

Запрос может формулироваться в свободной форме. Например, «Мне нужен отчет по сумме процентов вклада. Как его получить?». То есть пользователь просит бота в свободной форме найти ему требуемые данные.

В качестве примеров именований отчетов можно взять формы статистической налоговой отчетности с сайта https://www.nalog.gov.ru/rn77/related_activities/statistics_and_analytics/forms/

Отметим, что пользователь не знает точное название отчета, и задача программы именно найти наиболее подходящее совпадение.

- Количество выдаваемых пользователю строк нужно определить, руководствуясь здравым смыслом.
- Поиск отчетов происходит по файлу CSV формата, разделителем в котором является символ “|”
- Порядок строк в файле — несколько сотен.

Пример файла

```
f91b806e-806f-4717-be82-555a56f7135e | ndfl2_year | Справка о доходах компании в разрезе года
80e0d395-5ac0-4ac4-9fb2-6431936d178e | clients1 | Список клиентов в разрезе регионов
40e8b16d-adf5-4f46-9d74-17fe92b9df28 | employees | Число работников компании в разрезе регионов РФ
```

При запуске программы указываются следующие параметры:
--data path-to-csv.csv — путь до файла CSV, разделитель в котором знак “|”,
--input-file input-path-to-file.txt — путь до файла с входными строками поиска.

Формат — обычный текст. Каждая строка — текст, по которому программа должна выполнить поиск. Пример содержимого файла:

```
Как получить отчет по суммам налогового вычета за 2023 год?
Где найти агрегированный отчет по вкладам по Свердловской области?
```

--output-file output-path-to-file.json — путь до файла с результатами поиска. Если файла нет, должен создаться, если есть, перезаписаться. Формат файла — json, содержащий следующие поля:

1. initTime — число, время в миллисекундах инициализации от начала запуска программы до готовности к выполнению первого поиска. Может включать в себя в том числе вычитку файла --input-file.
2. result — массив, каждый элемент которого является результатом поиска по строке файла --input-file. У объектов массива есть следующие поля:
   a) search — строка поиска;
   b) result — массив guid строк, подходящих под поиск, отсортированных по релевантности;
   c) time — число в миллисекундах, затраченное на выполнения поиска по строке.

Пример содержимого файла:

```
{“initTime”:100, “result”:[
    {“search”: ”Как получить отчет по суммам налогового вычета за 2023 год?”, “result”: [“1a9f7664-9362-4f16-9a9d-7242c0cca6d3”, “03066ad5-b213-420e-9273-f8018a3bc9b0”], “time”: 10},
    {“search”: ”Где найти аггрегированный отчет по вкладам по Свердловской области?”, “result”: [“442f1a87-3a93-4670-95a7-aed4a857dc0e”], “time”: 10}
]}
```

- Проверка на ряд условий осуществляется автоматически, на сдачу задания дается 3 попытки.
- Учитывается лучший результат.
- Проверка происходит на linux системах.

Процесс проверки

1. Сборка: mvn clean package
2. Копирование артефакта search-*.jar из папки target в директорию проверки.
3. Автоматизированный запуск артефакта под различные критерии проверки: java -jar search.jar --data /data/test.csv --input-file /temp/input1.txt --output-file /temp/result1.json
4. Проверка результатов из файлов результатов, которые были созданы приложением.

Нефункциональные требования

1. Нельзя использовать сторонние отдельно стоящие сервисы, а также сервисы сети интернет.
2. Должны соблюдаться принципы ООП и SOLID.
3. Ошибочные и краевые ситуации должны быть корректно обработаны.
4. Решение должно быть основано на алгоритмах без использования машинного обучения и нейронных сетей.

Основное внимание будет уделяться точности предложенных вариантов. Также требуется в файле README.MD предоставить оценку сложности реализованного алгоритма.
В случае, если возникает вопрос, который не покрывает данная постановка задачи, выбирайте любое его решение не противоречащее постановке. В readme должен быть отражен вопрос и принятое решение.