# ИНФОПОИСК И БД: проект психологической помощи

 Выполнила: </n>

Захарова Яна

# ИНФОПОИСК

# 1. Корпус и предобработка

1. Корпус вопросов к психологу и ответов от психологов с сайта psiholog.ru
2. tf-idf - нижний регистр, удаление пунктуации, лемматизация, удаление стоп-слов
3. bert - BertTokenizer

# 2. Два способа индексирования

1. **TF-IDF индексация:**

   TfidfVectorizer для создания матрицы TF-IDF. Она превращает тексты в числовые векторы, учитывая частоту слов в документе и их редкость в корпусе.
   Сохранение модели и самой матрицы происходит с помощью модуля pickle, чтобы при каждом запуске программы индексация не пересчитывалась заново.
2. **ruBERT индексация:**
   Использую ruBERT (DeepPavlov/rubert-base-cased), которая хорошо обучена на русскоязычных текстах.

   Текст токенизируется, BERT принимает токенизированные тексты и возвращает эмбеддинги.
   Эмбеддинги сохраняются в файл с помощью библиотеки pickle, чтобы их можно было загрузить при повторном запуске программы.

# 2. Особенности поиска

**На вход:**

- запрос
- тип индексации (tf-idf, bert)
- топ-х документов

**На выход:**

```
Используем BERT для поиска...
```

```
Поиск занял 3.4785 секунд.

Топ-2 подходящих документов:
Сходство: 0.9848
Вопрос: Дочь хочет уйти от мужа и оставить ему сына.
Ответ: ['Здравствуйте, Елена! Честно говоря, Вы ниче

Сходство: 0.9847
Вопрос: Лечение от компьютерной зависимости
Ответ: ['Марина, здравствуйте! Я Вас понимаю. Почти
```

# 3. Реализация API для поисковой системы

Разработано веб-приложение с использованием Flask:

- Приложение предоставляет доступ к методам индексации и поиска.
- Использованы Pydantic-модели для валидации данных:
- Определены структуры запросов и ответов, обеспечивая строгую проверку входных данных.

Реализованы основные эндпоинты:

- /methods (GET): Возвращает список доступных методов индексации (TF-IDF и BERT).
- /corpora (GET): Возвращает информацию о корпусе данных (количество примероа и название корпуса).
- /search (POST): Выполняет поиск по запросу с использованием выбранного метода индексации.

# 3. Реализация API для поисковой системы

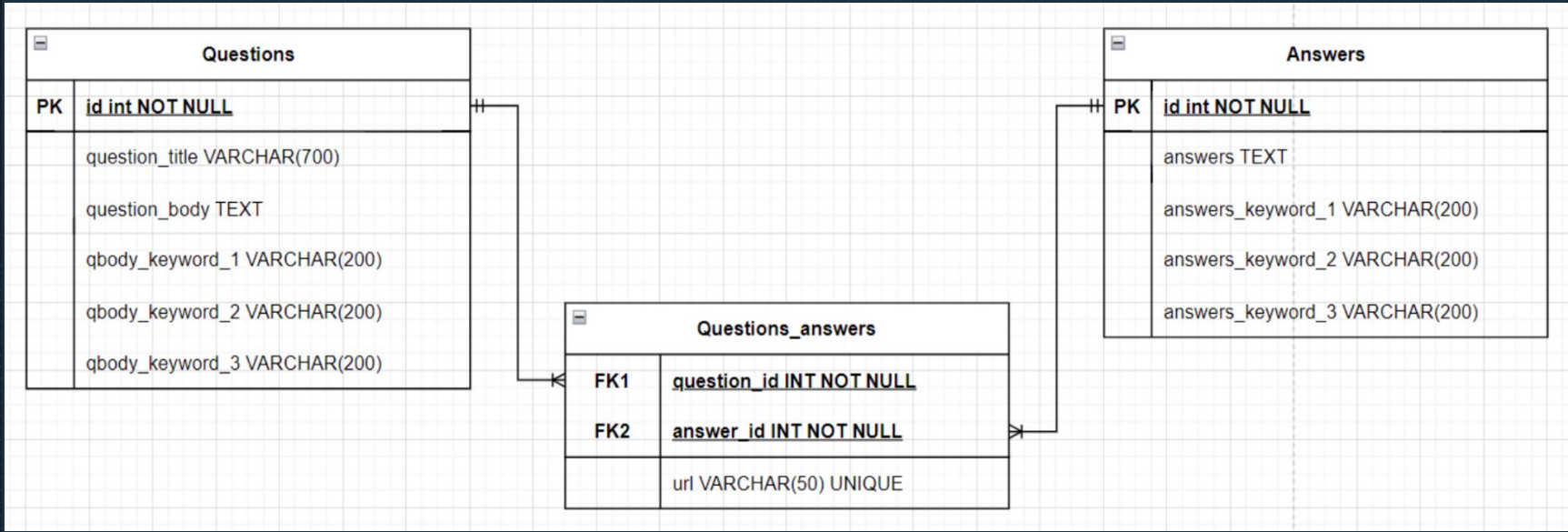
Реализована логика поиска через search_engine:

- Подключение к модулю для обработки запросов поиска.
- Поиск реализован на основе косинусного сходства (TF-IDF и BERT).

- Все входные данные проверяются с использованием Pydantic, что минимизирует риск ошибок.
- Включен режим debug, что позволяет быстро находить и устранять ошибки в процессе разработки.

# БД

# 1. Схема БД



# 2. Подключение к бд через Python

```python
import mysql.connector
from mysql.connector import connect, Error
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="T8xXSnKJ",
    database="psychologist_db"
)

cursor = connection.cursor()
cursor.execute("SELECT * FROM Questions;")
existing_data = cursor.fetchall()

for row in existing_data:
    print(row)

connection.close()
```

Python

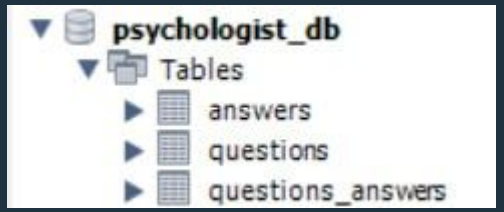
```
(1, 'Тест от психолога', 'спрашивает: Нина (неуказан)Здравствуйте! Для чего психолог предложил \nдочери представить себя посудой, растением, оружием, украшением
(2, 'Помогите разобраться', 'спрашивает: Евгений (неуказан)Здравствуйте.Меня зовут Евгений.Помогите пожалуйста разобраться в ситуации. \n   Я женат уже 13 лет, 
```

# 3. Добавление, удаление и обновление данных

1. Создание БД:
```
create_db_query = "CREATE DATABASE IF NOT EXISTS
psychologist_db;"
```
2. Создание таблиц по схеме:
```
create_questions_table = """
        CREATE TABLE IF NOT EXISTS Questions (
            id INT AUTO_INCREMENT PRIMARY KEY,
            question_title VARCHAR(700),
            question_body TEXT,
            qbody_keyword_1 VARCHAR(200),
            qbody_keyword_2 VARCHAR(200),
            qbody_keyword_3 VARCHAR(200)
        );
    """
```

# 3. Добавление, удаление и обновление данных

3. Создание прототипов таблиц в Pandas
4. Запись таблиц из Pandas в файл
5. Запись данных из файла в таблицы из БД



# 3. Добавление, удаление и обновление данных

## 6. Удаление из БД

```python
from mysql.connector import connect, Error

try:
    # Подключаемся к базе данных
    connection = connect(
        host="localhost",
        user="root",
        password="T8xXSnKJ",
        database="psychologist_db"
    )
    print("Соединение успешно установлено")

    # запрос для удаления строки
    delete_query = "DELETE FROM Questions WHERE id = %s;"
    record_id = 777

    with connection.cursor() as cursor:
        cursor.execute(delete_query, (record_id,))
        connection.commit()
        print(f"Строка с id={record_id} успешно удалена")

except Error as e:
    print(f"Ошибка: {e}")

finally:
    if connection and connection.is_connected():
        connection.close()
        print("Соединение закрыто")
```

```
Соединение успешно установлено
Строка с id=777 успешно удалена
Соединение закрыто
```

## 7. Добавление в БД

```python
try:
    connection = connect(
        host="localhost",
        user="root",
        password="T8xXSnKJ",
        database="psychologist_db"
    )
    print("Соединение успешно установлено")

    insert_query = """
        INSERT INTO Questions (id, question_title, question_body, qbody_keyword_1, qbody_keyword_2, qbody_keyword_3)
        VALUES (%s, %s, %s, %s, %s, %s);
    """
    record_data = (
        777,
        'как выжить в сессию',
        'Я учусь на 4м курсе, работаю, и сейчас сдаю сессию. Сессия очень сложная, проект на проекте, как мне выжить?',
        'сессия',
        'проект',
        'работать'
    )

    with connection.cursor() as cursor:
        cursor.execute(insert_query, record_data)
        connection.commit()
        print(f"Строка с id={record_data[0]} успешно добавлена")
```

```
Соединение успешно установлено
Строка с id=777 успешно добавлена
Соединение закрыто
```

# 3. Добавление, удаление и обновление данных

8. Обновление БД

```
update_query = """
UPDATE Questions
SET
    question_title = 'Как справиться с эмоциональным выгоранием?',
    question_body = 'Я чувствую себя постоянно уставшим и подавленным. Как мне восстановить силы?',
    qbody_keyword_1 = 'выгорание',
    qbody_keyword_2 = 'усталость',
    qbody_keyword_3 = 'восстановление'
WHERE id = 777;  -- Условие обновления
"""
```

# 4. Запросы

Простой запрос:

```
easy = """
            SELECT q.question_title, q.question_body, a.answers
            FROM Questions q
            JOIN questions_answers q_a on q.id = q_a.question_id
            JOIN Answers a ON q_a.answer_id = a.id
            WHERE qbody_keyword_1 = 'депрессия'
           or qbody_keyword_2 = 'смерть'
           or qbody_keyword_3 = 'работа';
            """
```

# 4. Запросы

Запрос с агрегированной функцией и сортировкой:

```
medium = """
            SELECT count(qbody_keyword_1) as summa_plus
            FROM Questions q
            JOIN questions_answers q_a on q.id = q_a.question_id
            JOIN Answers a ON q_a.answer_id = a.id
            where qbody_keyword_1 in ('любовь', 'солнце', 'счастье',
'свет', 'жизнь')
            """
```

# 4. Запросы

Запрос с агрегированной функцией и группировкой:

```
medium = """
            SELECT qbody_keyword_2, count(qbody_keyword_2) as
summa_q_theme,
            answers_keyword_1, count(answers_keyword_1) as
summa_a_theme
            FROM Questions q
            JOIN questions_answers q_a on q.id = q_a.question_id
            JOIN Answers a ON q_a.answer_id = a.id
            group by qbody_keyword_2, answers_keyword_1
            order by count(qbody_keyword_2) desc,
count(answers_keyword_1) desc
            limit 100
            """
```

# 5. Chroma

1. Создана коллекция из моего датасета

```
100%|██████████| 9933/9933 [1:10:18<00:00,  2.35it/s]
```

2. Можно выполнять запросы

```
# Выполнение запроса
query_text = "Как научиться снова доверять жене?"
results = collection.query(
    query_texts=[query_text],
    n_results=5
)

print("Результаты поиска:", results)
```

✓ 0.3s

```
Результаты поиска: {'ids': [['doc_6506', 'doc_7718', 'doc_2131', 'doc_6505', 'doc_3250']],
```