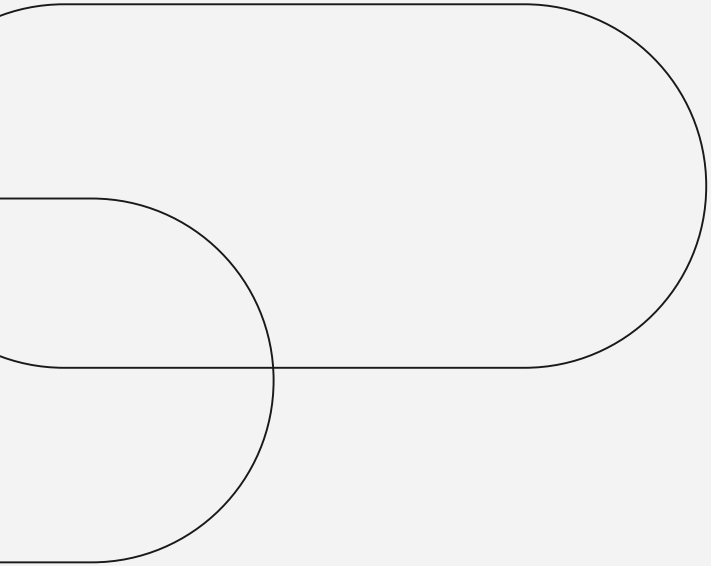


FastAPI

Application Programming Interface

01 Jupyter

Подключение

NLP

Подключение библиотек

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import pandas as pd
import nltk
import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

Загрузка данных

[3]:
```
df = pd.read_csv('dataset03.csv')
```

[4]:
```
df
```

[4]:

	namecompany	description	rating	activity	date_publish	text_article	prep_text	tokenize_text	text_stem
0	Timeweb Cloud	Облачная платформа для разработчиков и бизнеса	1572.77	Веб-разработка, Домены и хостинг, Веб-сервисы	2023-03-27	В своём блоге я рассмотрел устройство и назнач...	в своём блоге я рассмотрел устройство и назнач...	своём блоге рассмотрел устройство назначение к...	сво блог рассмотрел устройств назначен контрол...
1	OTUS	Цифровые навыки от ведущих экспертов	795.43	Консалтинг и поддержка, Рекрутинг и HR, Произв...	2023-03-27	Автор статьи: Роман КозловРуководитель курса B...	автор статьи роман козловруководитель курса bi...	автор статьи роман козловруководитель курса bi...	автор стат рома козловруководител курс bi анал...
2	Онлайн Патент	Ваш личный патентный офис	220.74	Консалтинг и поддержка, Веб-сервисы	2023-03-27	Рынок биологически активных добавок(БАД) с каж...	рынок биологически активных добавок бад с кажд...	рынок биологически активных добавок бад каждым...	рынок биологическ активн добавок бад кажд год ...

Векторизация

Векторизация текстовых данных

[4]:
```
# загрузка списка стоп-слов
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HOME\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[4]:
```
True
```

[5]:
```
# загрузка стоп-слов на русском языке
russian_stopwords = stopwords.words("russian")
```

TF-IDF

TfidfVectorizer - это инструмент векторизации текста в машинном обучении, который используется для преобразования текстовых данных в числовые признаки. Он основан на понятии TF-IDF (Term Frequency-Inverse Document Frequency), который позволяет оценить важность слова в документе.

TF-IDF вычисляется для каждого слова в документе путем умножения его частоты встречаемости (TF) на обратную частоту документа (IDF). TF оценивает, насколько часто слово появляется в документе, в то время как IDF оценивает, насколько информативно слово является путем снижения веса слова, которое часто встречается во всей коллекции документов.

TfidfVectorizer векторизует текст, создавая матрицу, в которой строки представляют документы, а столбцы - каждое уникальное слово или термин. Значения в ячейках этой матрицы представляют собой значения TF-IDF для каждого термина в каждом документе. Таким образом, каждый документ представлен в виде вектора TF-IDF значений, который может быть использован в алгоритмах машинного обучения.

[6]:
```
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000,
                                   min_df=0.01, stop_words=russian_stopwords,
                                   ngram_range=(1,3))
tfidf_matrix = tfidf_vectorizer.fit_transform(df['text_lemm'])
```

[7]:
```
tfidf_matrix.shape
```

[7]:
```
(438, 10000)
```

[8]:
```
print(tfidf_matrix)
```

```
  (0, 8947)     0.017091908228614894
  (0, 2203)     0.018917742841662737
  (0, 1831)     0.01187840743372064
```

Кластеризация

[12]:
```
pip install threadpoolctl==3.1.0
```

```
Requirement already satisfied: threadpoolctl==3.1.0 in c:\users\home\anaconda3\lib\site-packages (3.1.0)
Note: you may need to restart the kernel to use updated packages.
```

Метод к-средних - KMeans

[13]:
```
from sklearn.cluster import KMeans
```

[14]:
```
# ввод количества кластеров
num_clusters = 5

# обучение модели к-средних
km = KMeans(n_clusters=num_clusters)
idx = km.fit(tfidf_matrix)

# список с значениями кластеров
clusters = km.labels_.tolist()
```

[15]:
```
frame = pd.DataFrame()

#k-means
out = { 'text': df['text_article'], 'cluster': clusters }
frame = pd.DataFrame(out, columns = ['text', 'cluster'])
```

[16]:
```
frame
```

[16]:

	text	cluster
0	В своём блоге я рассмотрел устройство и назнач...	0
1	Автор статьи: Роман КозловРуководитель курса В...	0
2	Рынок биологически активных добавок(БАД) с каж...	4
3	Автор статьи: Александр КолесниковВирусный ана...	1
4	Всем привет! Меня зовут Артём Пузанков, я DevS...	2

Анализ кластеров

[17]:
```
import gensim
```

[18]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HOME\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

[18]:
```
True
```

[19]:
```
def tokenize(text):
    t = word_tokenize(text)
#     получение списка токенов, исключая русские стоп-слова
    tokens = [token for token in t if token not in russian_stopwords]
    return tokens
```

[20]:
```
frame['text_lemm'] = df['text_lemm']
frame['text_lemm3'] = [tokenize(text) for text in frame['text_lemm']]
```

Облако слов

Облако слов (Word Cloud) - это визуализация, которая отображает наиболее часто встречающиеся слова в наборе текстовых данных. В облаке слов размер и положение каждого слова зависят от его частоты в тексте: чем чаще слово встречается, тем больше его размер и тем ближе оно к центру.

Облака слов часто применяются для визуализации результатов анализа текста, выявления ключевых тем и идей, а также для быстрого восприятия наиболее релевантных слов в текстовых данных.

[21]:
```
!pip install WordCloud
```

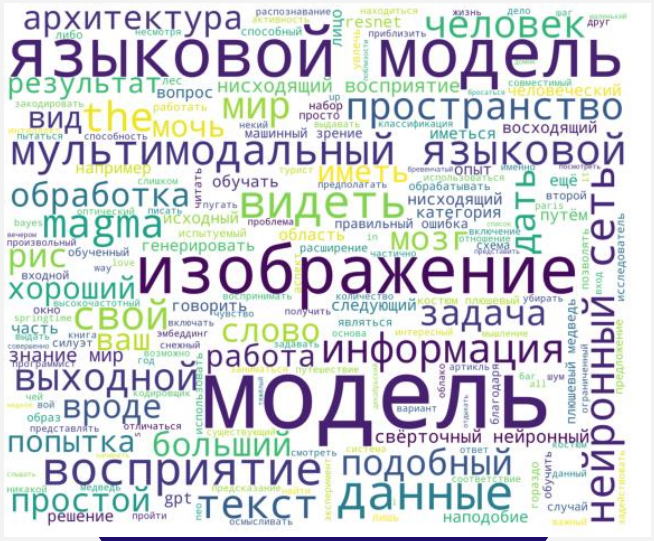
Анализ кластеров

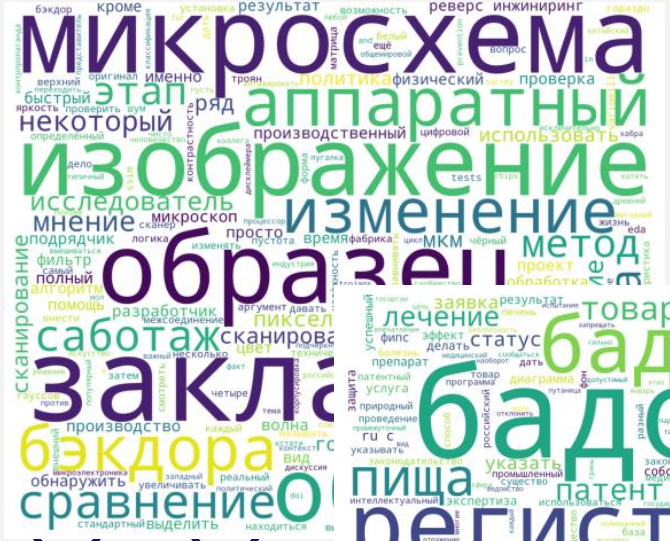
```
from wordcloud import (
    WordCloud,
    STOPWORDS
)
```

```
# получение текстовой строки из списка слов
def str_corpus(corpus):
    str_corpus = ''
    for i in corpus:
        str_corpus += ' ' + i
    str_corpus = str_corpus.strip()
    return str_corpus

# получение списка всех слов в корпусе
def get_corpus(data):
    corpus = []
    for phrase in data:
        for word in phrase.split():
            corpus.append(word)
    return corpus

# получение облака слов
def get_wordCloud(corpus):
    wordCloud = WordCloud(background_color='white',
                          stopwords=russian_stopwords,
                          width=3000,
                          height=2500,
                          max_words=200,
                          random_state=42
                         ).generate(str_corpus(corpus))
    return wordCloud
```

```
num = 0
for i in frame[frame['cluster']==0]['text_lemm3']:
    num += 1
    corpus = get_corpus(i)
    procWordCloud = get_wordCloud(corpus)

    fig = plt.figure(figsize=(17, 6))
    plt.subplot(1, 2, 1)
    plt.imshow(procWordCloud)
    plt.axis('off')
    plt.show()
    if num == 5:
        break
```

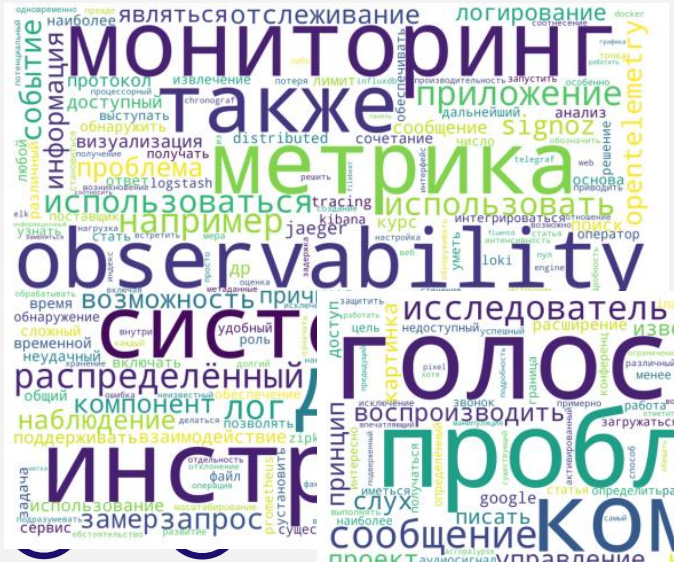
Кластер 0

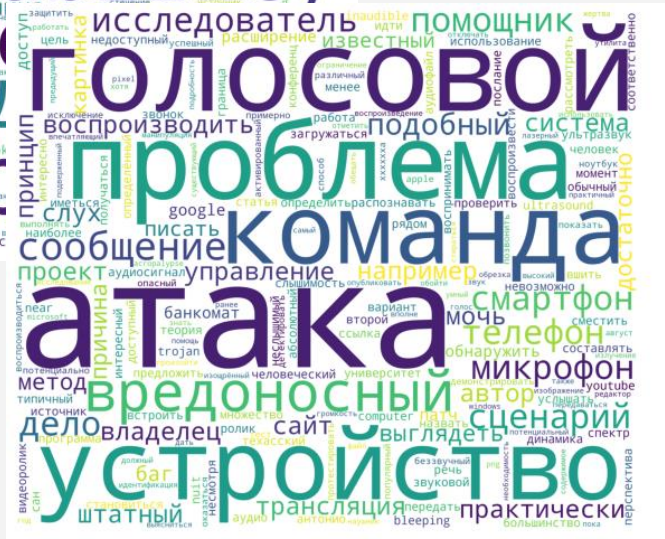




Статьи, связанные с исследованием каких либо изобретений, моделей.

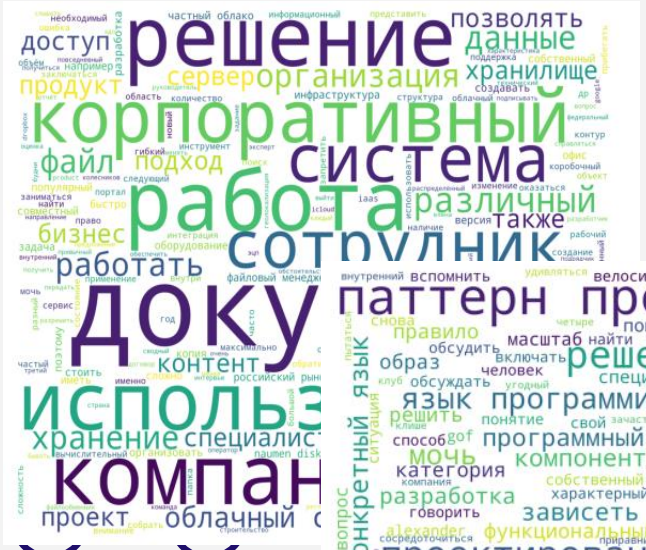
Кластер 1





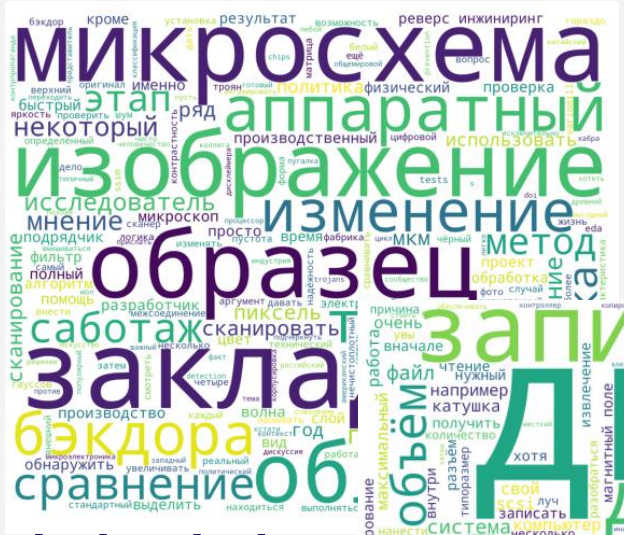
Статьи, связанные с системами, их использованием. Рассказываются какие либо проблемы и как их решить

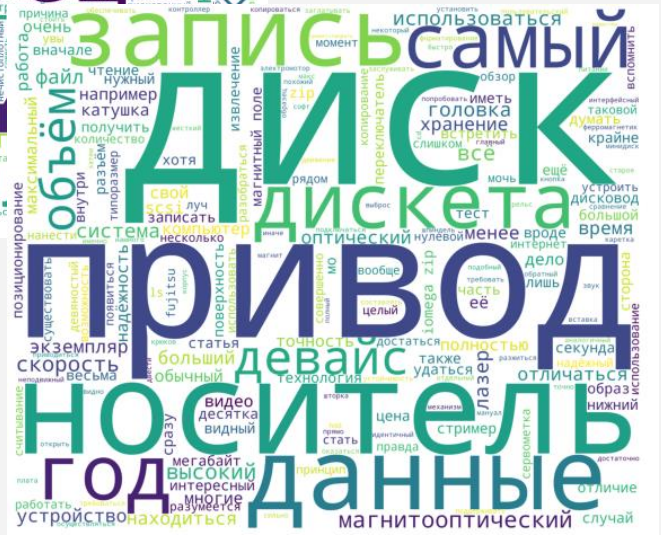
Кластер 2



Статьи, связанные с обзором данных и с разработкой проектов для анализа данных

Кластер 3





Статьи, связанные с устройствами, контролерами такие как ноутбук, смартфон, привод, микрофон. Исследования данных устройств.

Кластер 4

Статья содержит много кода либо англоязычных слов.

Классификация

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

```
# разделение на две выборки датафрейма с кластерами
X_train, X_test, y_train, y_test = train_test_split(frame['text'], frame['cluster'], test_size=0.25, random_state=42)
```

```
# преобразование текстовых данных в числовые признаки с помощью TF-IDF
vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1, 2), stop_words=russian_stopwords)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

LogisticRegression

```
# обучение модели логистической регрессии
model_lr = LogisticRegression()
model_lr.fit(X_train_tfidf, y_train)

# предсказание на тестовом наборе
y_pred = model_lr.predict(X_test_tfidf)

# оценка качества модели
accuracy_lr = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_lr)
```

```
Accuracy: 0.7636363636363637
```

RandomForestClassifier

[30]:
```
# обучение модели случайного леса
model_rf = RandomForestClassifier()
model_rf.fit(X_train_tfidf, y_train)

# предсказание на тестовом наборе
y_pred = model_rf.predict(X_test_tfidf)

# оценка качества модели
accuracy_rf = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_rf)
```

```
Accuracy: 0.7363636363636363
```

KNeighborsClassifier

[32]:
```
# обучение модели k-ближайших соседей
model_knn = KNeighborsClassifier()
model_knn.fit(X_train_tfidf, y_train)

# предсказание на тестовом наборе
y_pred = model_knn.predict(X_test_tfidf)

# оценка качества модели
accuracy_knn = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_knn)
```

```
Accuracy: 0.7272727272727273
```

[33]:
```
print("Точность логистической регрессии -", accuracy_lr)
print("Точность случайного леса -", accuracy_rf)
print("Точность дерева решений -", accuracy_dt)
print("Точность k-ближайших соседей -", accuracy_knn)
```

```
Точность логистической регрессии - 0.7636363636363637
Точность случайного леса - 0.7363636363636363
Точность дерева решений - 0.5181818181818182
Точность k-ближайших соседей - 0.7272727272727273
```

Сохранение моделей

```
import pickle

with open('model_rf.pkl', 'wb') as file:
    pickle.dump(model_rf, file)
```

```
with open('vectorizer.pkl', 'wb') as f:
    pickle.dump(vectorizer, f)
```

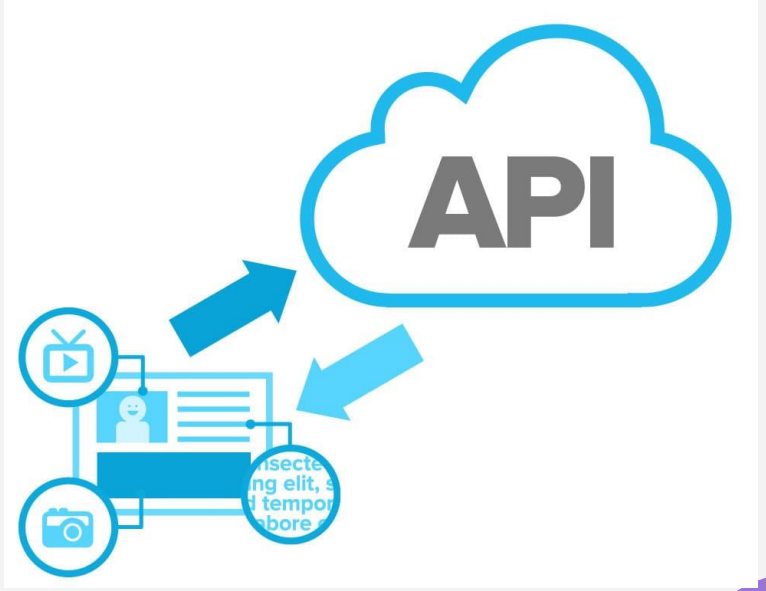
```
frame.to_csv("Data_NLP7.csv")
```

02
FastAPI

Кто такой FastAPI?

API (Application Programming Interface) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

FastAPI — веб-фреймворк для создания API, написанный на Python. Один из самых быстрых[2] и популярных (после Django и Flask)[3] веб-фреймворков, написанных на Python (на 2023 год).



Подключение FastAPI

```
(venv) C:\Users\Admin\PycharmProjects\api_text_clust>pip install fastapi[all]
```

```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
```

Библиотеки и создание API

Middleware (промежуточное или связующее программное обеспечение) — это фрагмент кода в конвейере приложения, используемый для обработки запросов и ответов.

```python
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
import pickle
import numpy as np
import pandas as pd
from pydantic import BaseModel
import string
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import pymorphy2
```

```python
# создания апи
app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],)

np.set_printoptions(suppress=True)
```

Загрузка моделей

```
# загрузка моделей
with open('model_rf.pkl', 'rb') as file:
    model = pickle.load(file)

with open('vectorizer.pkl', 'rb') as file:
    vectorizer = pickle.load(file)

with open('text2.txt', 'r', encoding='utf-8') as file:
    text2 = file.read()
```

Функции по обработке текста

```python
# преведение текста к нижнему регистру без знаков пунктуации
def fun_punctuation_text(text):
    text = text.lower()
    text = ''.join([ch for ch in text if ch not in string.punctuation])
    text = ''.join([i if not i.isdigit() else '' for i in text])
    text = ''.join([i if i.isalpha() else ' ' for i in text])
    text = re.sub(r'\s+', ' ', text, flags=re.I)
    text = re.sub('[a-z]', '', text, flags=re.I)
    st = '\xa0'
    text = ''.join([ch if ch not in st else ' ' for ch in text])
    return text


# токенизация текста
def fun_tokenize_text(text):
    nltk.download('stopwords')
    stopword_eng = stopwords.words("english")
    stopword_ru = nltk.corpus.stopwords.words('russian')
    stopword_ru.extend(['т.д.', 'т', 'д', 'это', 'который', 'с', 'своём', 'всем', 'мб'])

    nltk.download('word_tokenize')
    nltk.download('punkt')
    t = word_tokenize(text)
    text = [token for token in t if token not in stopword_eng and token not in stopword_ru]
    return text
```

```python
# предобработка теста.
def fun_pred_text(text):
    text = fun_punctuation_text(text)
    text = fun_tokenize_text(text)
    text = fun_lemmatizing_text(text)
    return text
```

```python
# лемматизация текста
def fun_lemmatizing_text(text):
    morph = pymorphy2.MorphAnalyzer()
    res = list()
    for word in text:
        p = morph.parse(word)[0]
        res.append(p.normal_form)
    return res
```

Функции для предсказания

```python
# TF-IDF
def fun_TfidfVectorizer(text):
    text = fun_pred_text(text)

    df = pd.DataFrame({'text': str_corpus(text)}, index=range(1))
    new_row = pd.DataFrame({'text': text2}, index=range(1))
    df = pd.concat([df, new_row])

    tfidf_matrix = vectorizer.transform(df['text'])
    return tfidf_matrix
```

```python
# преобразование списка в текст
def str_corpus(corpus):
    str_corpus = ''
    for i in corpus:
        str_corpus += ' ' + i
    str_corpus = str_corpus.strip()
    return str_corpus
```

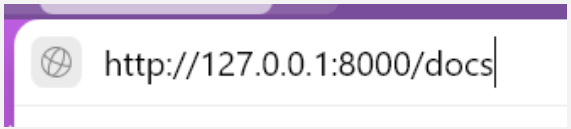
```python
# функция предсказывает кластер
def fun_predict_cluster(text):
    matrix = fun_TfidfVectorizer(text)
    predict_cluster = model.predict(matrix)
    lst = predict_cluster.tolist()
    mapping = {
        0: ' 0 - статьи, связанные с исследованием каких либо изобретений, моделей.',
        1: ' 1 - статьи, связанные с системами, их использованием. Рассказываются какие либо проблемы и как их решить.',
        2: ' 2 - статьи, связанные с обзором данных и с разработкой проектов для анализа данных.',
        3: ' 3 - статьи, связанные с устройствами, контролерами такие как ноутбук, смартфон, привод, микрофон. Исследования данных устройств.',
        4: ' 4 - статьи содержит много кода либо англоязычных слов.'
    }
    selected_cluster = mapping[lst[0]]
    return selected_cluster
```

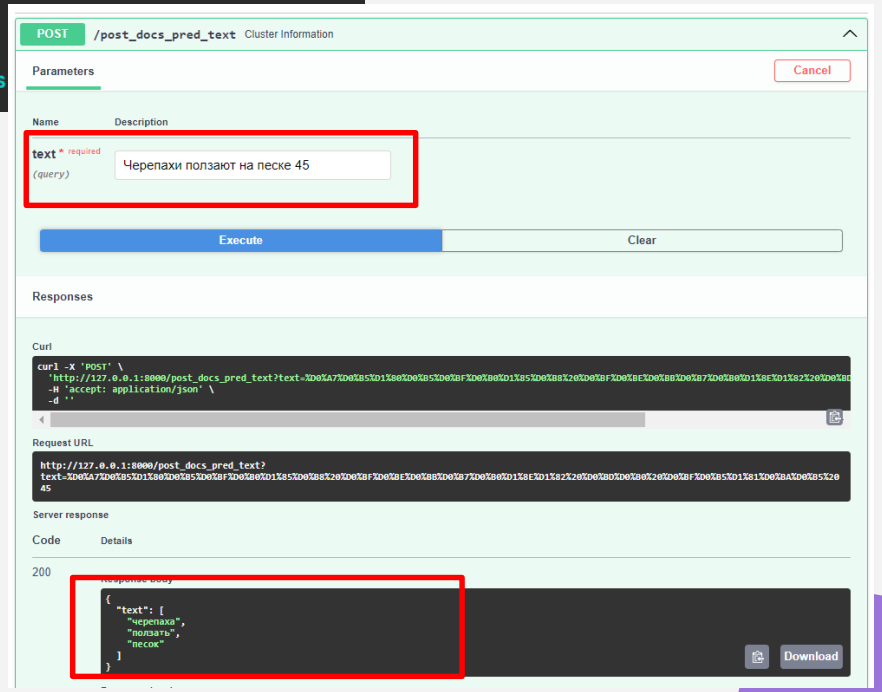
Получение и отправка данных

```python
# метод для docs
@app.post("/post_docs_pred_text")
def cluster_information(text):
    return {'text': fun_pred_text(text)}


class Item(BaseModel):
    text: str


# метод для подключения к интерфейсу
@app.post("/post_cluster")
def post_pred_text(item: Item):
    return {'cluster': fun_predict_cluster(item.text)}
```

Запуск API

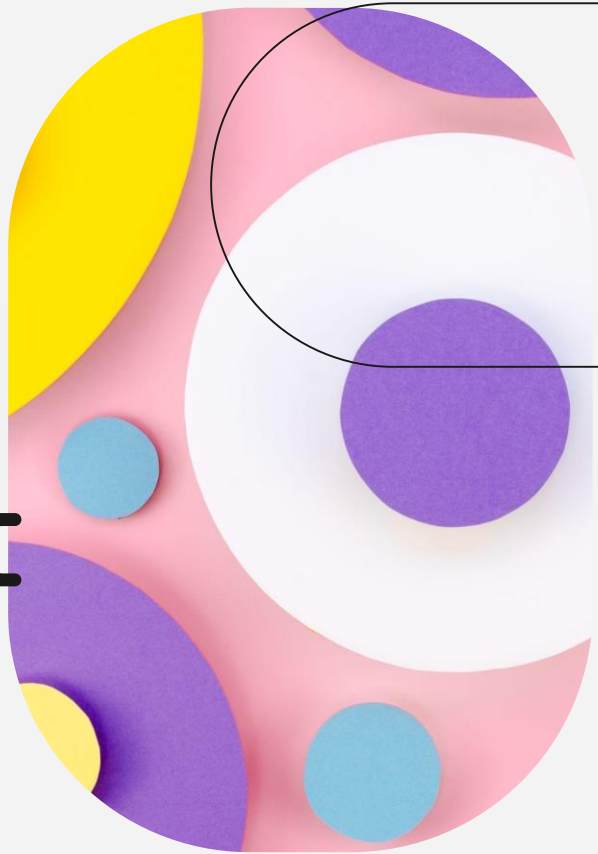
```
(venv) C:\Users\Admin\PycharmProjects\api_text_clust>uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\Users\\Admin\\Pycha
rmProjects\\api_text_clust']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press
INFO:     Started reloader process [6140] using WatchFiles
```



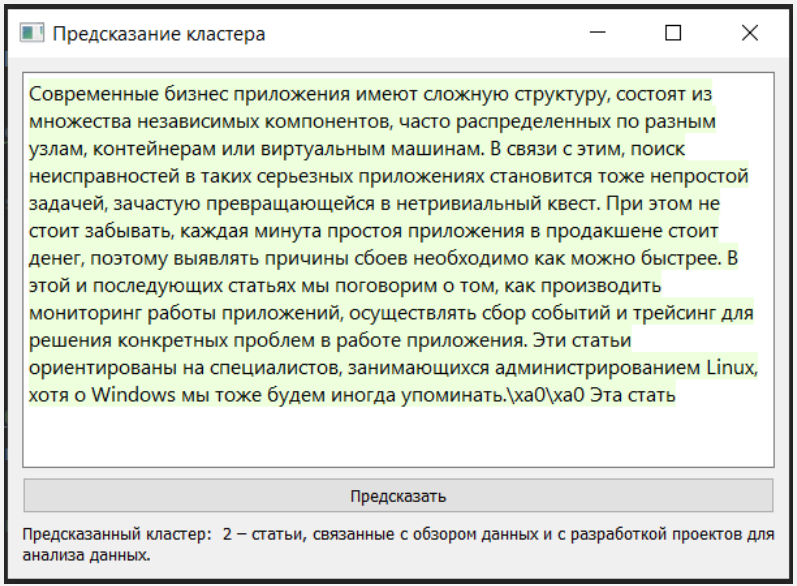


Интерактивную документацию по API

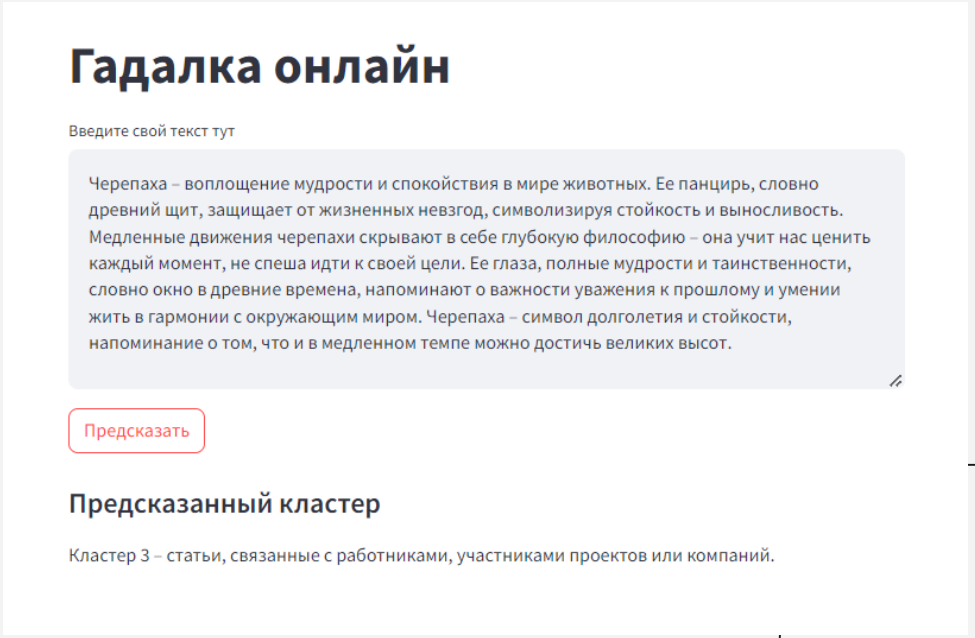
03
Интерфейс



PyQt



Streamlit



PyQt импорт

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout, \
    QHBoxLayout, QLineEdit, QPushButton, QLabel, \
    QWidget, QTextEdit
import requests
```

Создание окна

```python
# Класс для шаблона окна нашего приложения
class PredictionApp(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Предсказание кластера")
        self.setGeometry(100, 100, 600, 400)

        # Создание лейаута
        main_layout = QVBoxLayout()

        # Создание поля для ввода
        self.input_field = QTextEdit()
        self.input_field.setPlaceholderText("Ввод текста")
        main_layout.addWidget(self.input_field)

        # Создание кнопки предсказывания
        self.predict_button = QPushButton("Предсказать")
        self.predict_button.clicked.connect(self.predict_cluster)
        main_layout.addWidget(self.predict_button)
```

```python
# Создание лейбла для вывода ответа
self.output_label = QLabel("Нажмите предсказать для отображения кластера")
self.output_label.setWordWrap(True)
main_layout.addWidget(self.output_label)

# Создание главного лейаута для приложения
central_widget = QWidget()
central_widget.setLayout(main_layout)
self.setCentralWidget(central_widget)
```

Предсказание кластера

```python
def predict_cluster(self):
    # Получаем текст из нашего поля
    text = self.input_field.toPlainText()

    # Вызываем APIшку для предсказывания кластера
    data = {
        "text": text
    }

    url = "http://127.0.0.1:8000/post_cluster"
    response = requests.post(url, json=data)
    result = response.json()
    clust = result.get("cluster")

    # Выводим предсказанный кластер
    self.output_label.setText(f"Предсказанный кластер: {clust}")
```

Ну и главный метод который создаёт само приложение

```python
# Ну и главный метод который создаёт само приложение
if __name__ == "__main__":
    app = QApplication(sys.argv)
    prediction_app = PredictionApp()
    prediction_app.show()
    sys.exit(app.exec_())
```

Streamlit

```python
# Загрузка библиотек
import streamlit as st
import requests
import os


# Обнуление статусов прокси для корректного подключения
os.environ['HTTP_PROXY'] = ''
os.environ['HTTPS_PROXY'] = ''
```

```
(venv) C:\Users\Admin\PycharmProjects\api_text_clust>pip install streamlit
```

Создание всего

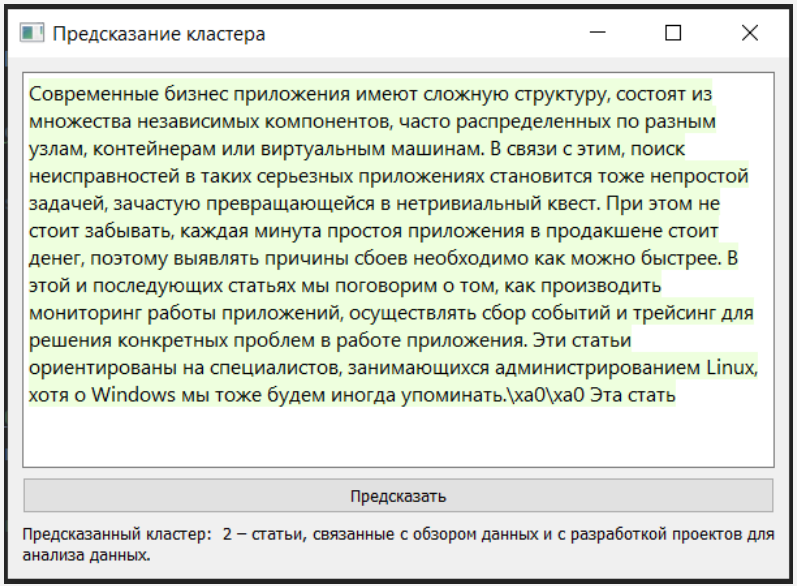
```python
def main():
    st.title("Гадалка онлайн")
    # Поле ввода
    input_text = st.text_area("Введите свой текст тут", height=200)

    # Кнопка предсказания
    if st.button("Предсказать"):
        if input_text == "":
            st.write(f"Введите текст")
        else:
            # Вызов АПИшки
            data = {
                "text": input_text
            }

            url = "http://127.0.0.1:8000/post_cluster"
            response = requests.post(url, json=data)
            result = response.json()
            clust = result.get("cluster")

            # Вывод предсказанного кластера
            st.markdown(f"""
                        #### Предсказанный кластер
                    """)
            st.write(f"Кластер {clust}")
```

```python
if __name__ == "__main__":
    main()
```

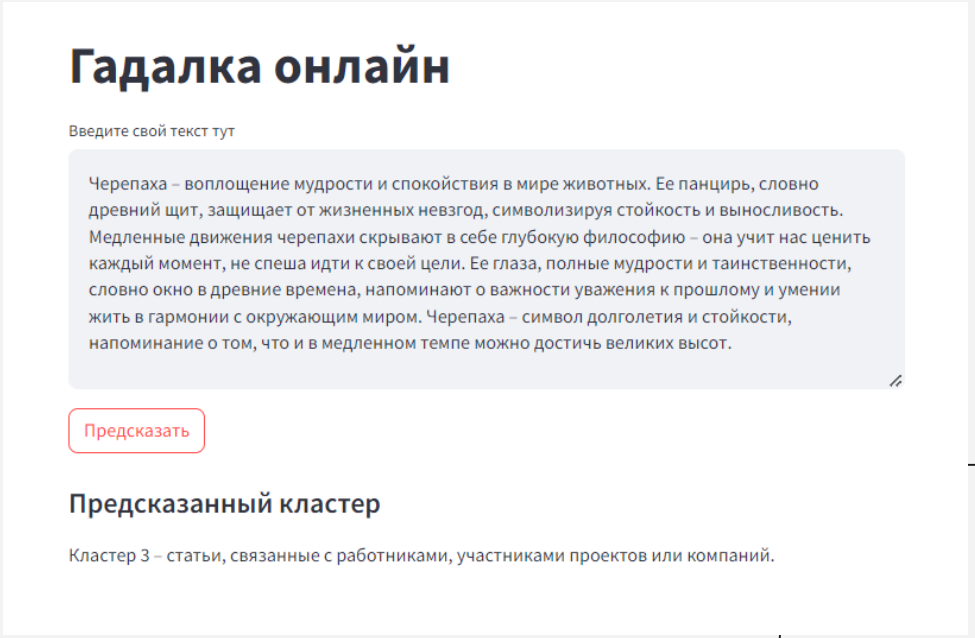
Запуск streamlit

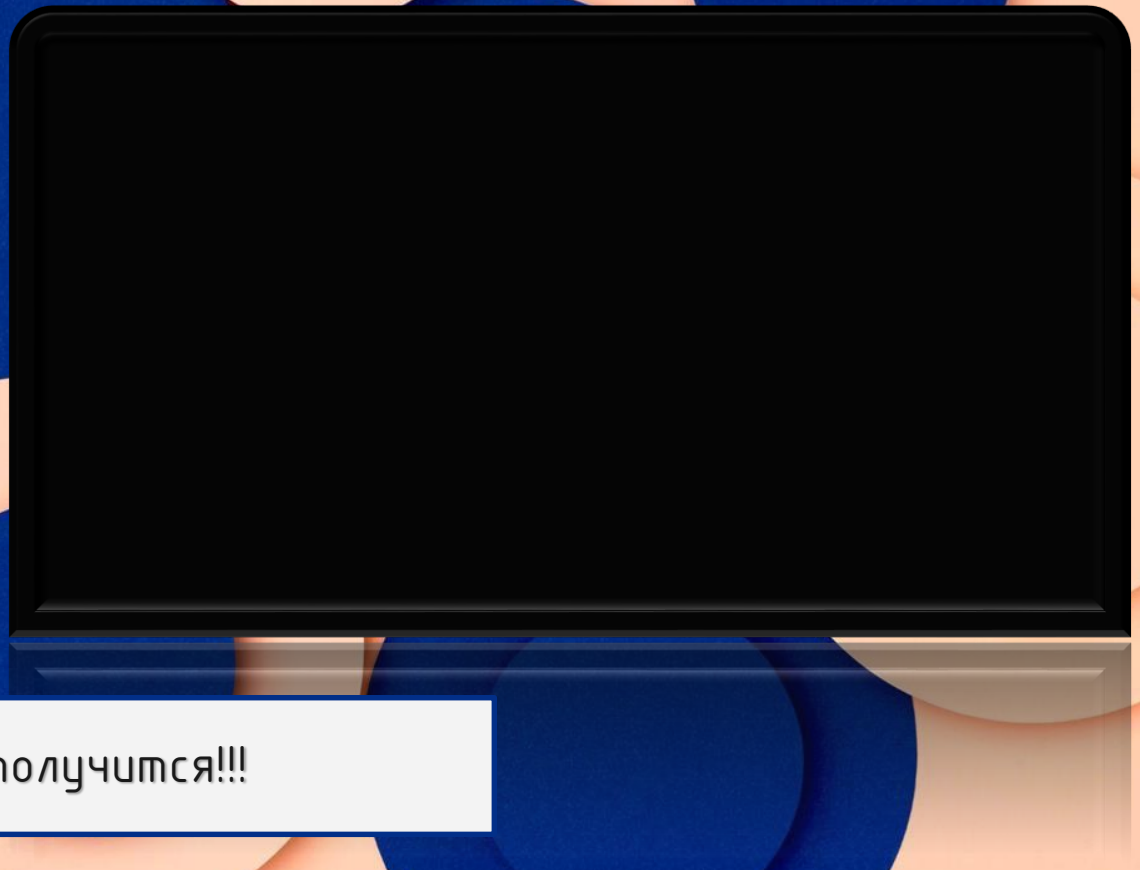
```
(venv) C:\Users\Admin\PycharmProjects\api_text_clust>streamlit run app_streamlit.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.98.249:8501
```

PyQt



Streamlit





У вас всё получится!!!