Программа предсказания статей

Сервер:

```python
from fastapi import FastAPI, HTTPException
import pickle
from pydantic import BaseModel
import string
import re
import nltk
from nltk.tokenize import word_tokenize
import pymorphy3

app = FastAPI()

try:
    with open('model_articles.pkl', 'rb') as file:
        model = pickle.load(file)

    with open('vectorizer_articles.pkl', 'rb') as file:
        vectorizer = pickle.load(file)
except Exception as e:
    raise RuntimeError(f"Ошибка загрузки моделей: {e}")

nltk.download('stopwords')
russian_stopwords = nltk.corpus.stopwords.words('russian')
russian_stopwords.extend(['т.д', 'это', 'который', 'с', 'своём', 'всем', 'свой', 'весь', 'привет', 'хабр'])


1 usage
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])
```

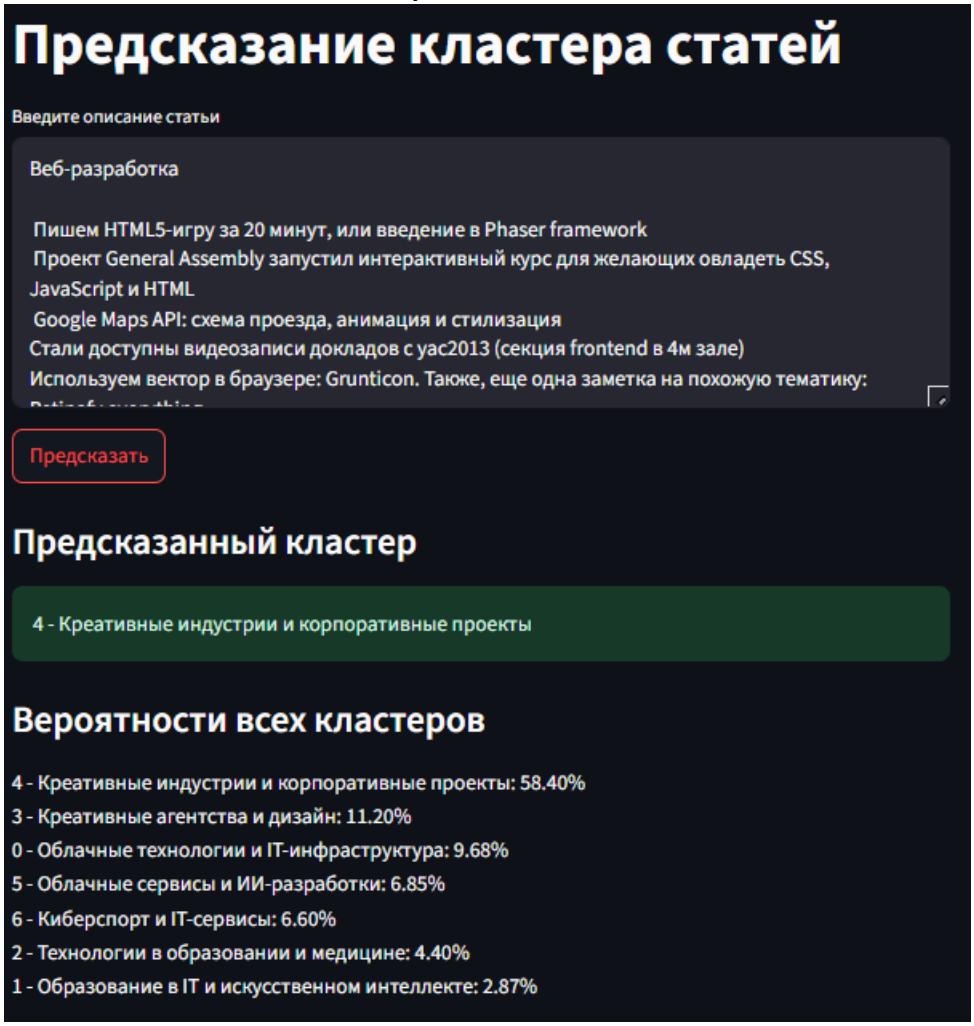
```
    text = re.sub( pattern: r'\s+', repl: ' ', text, flags=re.I)
    st = '❩\xa0'
    text = ''.join([ch if ch not in st else ' ' for ch in text])
    return text
1 usage
def fun_lemmatizing_text(text):
    tokens = word_tokenize(text)
    res = []
    morph = pymorphy3.MorphAnalyzer(lang='ru')
    for word in tokens:
        p = morph.parse(word)[0]
        res.append(p.normal_form)
    return " ".join(res)
1 usage
def fun_tokenize(text):
    t = word_tokenize(text)
    tokens = [token for token in t if token not in russian_stopwords]
    return " ".join(tokens)
1 usage
def fun_pred_text(text):
    text = fun_punctuation_text(text)
    text = fun_lemmatizing_text(text)
    text = fun_tokenize(text)
    return text
1 usage
def predict_cluster(text):
    cluster_description = {
        0: "0 - Облачные технологии и IT-инфраструктура",
        1: "1 - Образование в IT и искусственном интеллекте",
        2: "2 - Технологии в образовании и медицине",
        3: "3 - Креативные агентства и дизайн",
```

```
        3: "3 - Креативные агентства и дизайн",
        4: "4 - Креативные индустрии и корпоративные проекты",
        5: "5 - Облачные сервисы и ИИ-разработки",
        6: "6 - Киберспорт и IT-сервисы"
    }
    processed_text = fun_pred_text(text)
    text_vectorized = vectorizer.transform([processed_text])
    probabilities = model.predict_proba(text_vectorized)[0]
    sorted_clusters = sorted(
        [(cluster_description[i], prob) for i, prob in enumerate(probabilities)],
        key=lambda x: -x[1]
    )
    main_cluster = sorted_clusters[0][0]
    probabilities_str = "\n".join([f"{name}: {prob:.2%}" for name, prob in sorted_clusters])
    return main_cluster, probabilities_str
1 usage
class Item(BaseModel):
    text: str
@app.post("/predict")
def post_pred_text(item: Item):
    try:
        main_cluster, probabilities = predict_cluster(item.text)
        return {
            'cluster': main_cluster,
            'probabilities': probabilities
        }
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

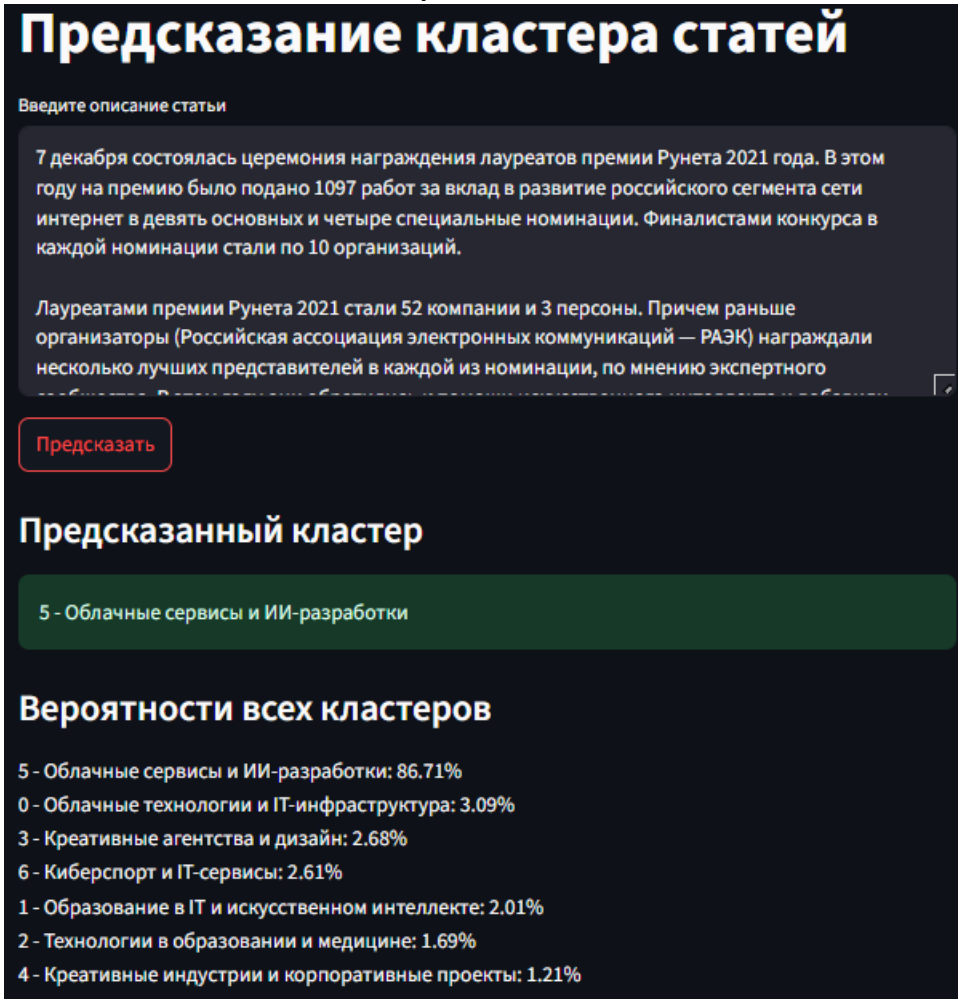
Клиент:

```python
import streamlit as st
import requests

st.set_page_config(page_title="Предсказание тем статей")

st.title("Предсказание кластера статей")
input_text = st.text_area("Введите описание статьи", height=200)

if st.button("Предсказать"):
    if not input_text.strip():
        st.warning("Пожалуйста, введите текст статьи")
    else:
        try:
            response = requests.post(
                url: "http://127.0.0.1:8000/predict",
                json={"text": input_text}
            )
            response.raise_for_status()
            result = response.json()

            st.subheader("Предсказанный кластер")
            st.success(result['cluster'])

            st.subheader("Вероятности всех кластеров")
            st.text(result['probabilities'])

        except requests.exceptions.RequestException as e:
            st.error(f"Ошибка при запросе к серверу: {e}")
        except Exception as e:
            st.error(f"Произошла ошибка: {e}")
```

Ввод статьи из PDF файла:



Ввод статьи из Json файла:



Ввод статьи из Хабр: