ДОДАТОК А

Код програми

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address

app = Flask(__name__)
CORS(app)
limiter = Limiter(key_func=get_remote_address)
limiter.init_app(app)

import torch
from transformers import RobertaTokenizer,
RobertaForSequenceClassification
import faiss
import numpy as np
import nltk
import requests
import re
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import WordNetLemmatizer

from tone import analyze_tone
from bias_det import detect_bias

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

tokenizer = RobertaTokenizer.from_pretrained("roberta-large-
mnli")
model =
RobertaForSequenceClassification.from_pretrained("roberta-
large-mnli")
model.eval()

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

dimension = 768
index = faiss.IndexFlatL2(dimension)

def preprocess_text(text):
    text = re.sub(r'\W+', ' ', text.lower())
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(t) for t in tokens if t not
in stop_words]
    return [' '.join(tokens)]
```

```
def get_embedding(text):
    inputs = tokenizer(text, return_tensors="pt",
truncation=True, max_length=512)
    with torch.no_grad():
        outputs = model.roberta(**inputs)
        embedding = outputs.last_hidden_state.mean(dim=1)
    return embedding.numpy()

NEWS_API_KEY = "ae34796103424bbd8805fa95c50b7ca5"
NEWS_API_URL = "https://newsapi.org/v2/everything"

def fetch_news(query, num_articles=20):
    params = {
        'q': query,
        'apiKey': NEWS_API_KEY,
        'language': 'en',
        'pageSize': num_articles
    }
    response = requests.get(NEWS_API_URL, params=params)
    articles = response.json().get('articles', [])
    return [article['content'] or article['description'] for
article in articles if article['content'] or
article['description']]

@app.route('/api/verify', methods=['POST'])
@limiter.limit("10 per minute")
def verify_statement():
    data = request.get_json()
    statement = data.get('statement', '')
    if not statement:
        return jsonify({'error': 'Твердження є
обов'язковим'}), 400

    news_texts = fetch_news(statement)
    sentences = []
    for text in news_texts:
        if not text:
            continue
        processed = preprocess_text(text)[0]
        embedding = get_embedding(processed)
        sents = sent_tokenize(text)
        for sent in sents:
            sent_processed = preprocess_text(sent)[0]
            sent_embed = get_embedding(sent_processed)
            sentences.append((sent, sent_embed))

    processed_statement = preprocess_text(statement)[0]
    statement_embedding = get_embedding(processed_statement)

    evidence_candidates = []
    for sent, sent_vec in sentences:
        sim = np.dot(statement_embedding, sent_vec.T) /
```

```
(np.linalg.norm(statement_embedding) *
np.linalg.norm(sent_vec))
        if sim >= 0.8:
            evidence_candidates.append((sent, sim.item(),
sent_vec))

    results = []
    for text, score, vec in evidence_candidates:
        inputs = tokenizer(statement, text,
return_tensors="pt", truncation=True, padding=True,
max_length=512)
        with torch.no_grad():
            logits = model(**inputs).logits
            probs = torch.softmax(logits, dim=1).numpy()[0]
        labels = ['contradiction', 'neutral', 'entailment']
        max_idx = np.argmax(probs)
        results.append({
            'text': text,
            'relation': labels[max_idx],
            'confidence': float(probs[max_idx]),
            'score': float(score)
        })

    entailment = sum(1 for r in results if r['relation'] ==
'entailment')
    contradiction = sum(1 for r in results if r['relation'] ==
'contradiction')
    total = entailment + contradiction

    if total == 0:
        final_label = 'непереконливо'
    elif entailment > contradiction:
        final_label = 'ймовірно істинно'
    elif contradiction > entailment:
        final_label = 'ймовірно хибно'
    else:
        final_label = 'непереконливо'

    top_evidence = sorted(results, key=lambda x:
x['confidence'], reverse=True)[:3]

    return jsonify({
        'statement': statement,
        'result': final_label,
        'evidence': top_evidence
    })

@app.route('/api/analyze-tone', methods=['POST'])
@limiter.limit("10 per minute")
def analyze_tone_endpoint():
    data = request.get_json()
    text = data.get('text', '')
    if not text:
```

```
        return jsonify({'error': 'Текст є обов’язковим'}), 400

    result = analyze_tone(text)
    return jsonify({
        'text': text,
        'tone': result['tone'],
        'confidence': result['confidence']
    })


@app.route('/api/detect-bias', methods=['POST'])
@limiter.limit("10 per minute")
def detect_bias_endpoint():
    data = request.get_json()
    text = data.get('text', '')
    if not text:
        return jsonify({'error': 'Текст є обов’язковим'}), 400

    result = detect_bias(text)
    return jsonify({
        'text': text,
        'bias': result['bias'],
        'confidence': result['confidence'],
        'labels': result['labels'],
        'scores': result['scores']
    })

if __name__ == '__main__':
    app.run(debug=True)
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	69 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Кочаров П.А.		22.06	Веб-застосунок для інтелектуального аналізу достовірності інформації, що передається в ЗМІ	Шифр групи	Код напр./спец.
Перевірив		Черненко М.В.				ІТШІ-21-4	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					