```
# Step 0: Install dependencies
!pip install -q torch==2.8.0 transformers==4.34.0 langdetect googletrans==4.0.0-rc1 spacy folium
!python -m spacy download en_core_web_sm
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.5/121.5 kB 6.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 981.5/981.5 kB 42.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.7/7.7 MB 95.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.1/55.1 kB 3.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.4/133.4 kB 10.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.6/42.6 kB 2.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 kB 3.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.0/65.0 kB 5.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.8/3.8 MB 68.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 295.0/295.0 kB 19.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 55.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.6/53.6 kB 3.9 MB/s eta 0:00:00
  Building wheel for googletrans (setup.py) ... done
  Building wheel for langdetect (setup.py) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sou
firebase-admin 6.9.0 requires httpx[http2]==0.28.1, but you have httpx 0.13.3 which is incompatible.
mcp 1.14.1 requires httpx>=0.27.1, but you have httpx 0.13.3 which is incompatible.
gradio-client 1.13.0 requires httpx>=0.24.1, but you have httpx 0.13.3 which is incompatible.
gradio-client 1.13.0 requires huggingface-hub>=0.19.3, but you have huggingface-hub 0.17.3 which is incompatible.
diffusers 0.35.1 requires huggingface-hub>=0.34.0, but you have huggingface-hub 0.17.3 which is incompatible.
google-genai 1.38.0 requires httpx<1.0.0,>=0.28.1, but you have httpx 0.13.3 which is incompatible.
gradio 5.46.0 requires httpx<1.0,>=0.24.1, but you have httpx 0.13.3 which is incompatible.
gradio 5.46.0 requires huggingface-hub<1.0,>=0.33.5, but you have huggingface-hub 0.17.3 which is incompatible.
sentence-transformers 5.1.0 requires huggingface-hub>=0.20.0, but you have huggingface-hub 0.17.3 which is incompatible.
sentence-transformers 5.1.0 requires transformers<5.0.0,>=4.41.0, but you have transformers 4.34.0 which is incompatible.
peft 0.17.1 requires huggingface_hub>=0.25.0, but you have huggingface-hub 0.17.3 which is incompatible.
datasets 4.0.0 requires huggingface-hub>=0.24.0, but you have huggingface-hub 0.17.3 which is incompatible.
accelerate 1.10.1 requires huggingface_hub>=0.21.0, but you have huggingface-hub 0.17.3 which is incompatible.
langsmith 0.4.28 requires httpx<1,>=0.23.0, but you have httpx 0.13.3 which is incompatible.
openai 1.108.0 requires httpx<1,>=0.23.0, but you have httpx 0.13.3 which is incompatible.
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any.wh
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 89.3 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do t ✦ by selecting the
```

```
'Restart kernel' or 'Restart runtime' option.
```

```
!pip install --upgrade huggingface_hub
```

```
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.12/dist-packages (0.17.3)
Collecting huggingface_hub
  Downloading huggingface_hub-0.35.1-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (3.19.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (2025.3.0)
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (2.32.4)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (4.67.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (4.15.0
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub) (1.1.10)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface_hub)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface_hub) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface_hub) (2.5.(
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface_hub) (2025
Downloading huggingface_hub-0.35.1-py3-none-any.whl (563 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 563.3/563.3 kB 17.5 MB/s eta 0:00:00
Installing collected packages: huggingface_hub
  Attempting uninstall: huggingface_hub
    Found existing installation: huggingface-hub 0.17.3
    Uninstalling huggingface-hub-0.17.3:
      Successfully uninstalled huggingface-hub-0.17.3
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sou
tokenizers 0.14.1 requires huggingface_hub<0.18,>=0.16.4, but you have huggingface-hub 0.35.1 which is incompatible.
gradio-client 1.13.0 requires httpx>=0.24.1, but you have httpx 0.13.3 which is incompatible.
gradio 5.46.0 requires httpx<1.0,>=0.24.1, but you have httpx 0.13.3 which is incompatible.
sentence-transformers 5.1.0 requires transformers<5.0.0,>=4.41.0, but you have transformers 4.34.0 which is incompatible.
Successfully installed huggingface_hub-0.35.1
```

```
# Step 1: Imports & device setup
import torch
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"⚡ Using device: {device}")

import re
from datetime import datetime
from typing import List, Dict
from transformers import pipeline
from langdetect import detect
```

```
from langdetect.lang_detect_exception import LangDetectException
from googletrans import Translator
import spacy
import folium

# Load spaCy for location extraction
nlp = spacy.load("en_core_web_sm")
```

```
⚡ Using device: cpu
/usr/local/lib/python3.12/dist-packages/transformers/utils/generic.py:311: FutureWarning: `torch.utils._pytree._register_pytree_nod
  torch.utils._pytree._register_pytree_node(
```

```
# Step 2: OceanHazardAI class
class OceanHazardAI:
    def __init__(self):
        print("🤖 Initializing Ocean Hazard AI Engine...")

        # Sentiment Analyzer
        print("📝 Loading sentiment analysis model...")
        self.sentiment_analyzer = pipeline(
            "sentiment-analysis",
            model="cardiffnlp/twitter-roberta-base-sentiment-latest",
            device=0 if device.type != "cpu" else -1,
            return_all_scores=True
        )

        # Hazard Classifier
        print("🎯 Loading hazard classification model...")
        self.classifier = pipeline(
            "zero-shot-classification",
            model="valhalla/distilbart-mnli-12-1",
            device=0 if device.type != "cpu" else -1
        )

        # Translator
        print("🌍 Loading translation service...")
        self.translator = Translator()

        # Hazard labels
        self.hazard_labels = [
            "tsunami", "high waves", "flooding", "storm surge",
```

```
            "coastal erosion", "dangerous currents", "cyclone",
            "abnormal tides", "sea level rise", "normal weather"
        ]

        # Urgency keywords
        self.urgency_keywords = {
            'high': ['emergency','danger','evacuate','urgent','help','disaster','massive','huge','destroy','death','खतरा','आपातकाल'
            'medium': ['concern','worry','alert','warning','rising','चिंता','सावधान','चेतावनी','கவலை','எச்சரிக்கை','ఆందోళన','హెచ్చరిక'],
            'low': ['normal','calm','peaceful','fine','good','सामान्य','शांत','சாதாரண','அமைதி','సాధారణ','శాంతి']
        }

        # Coastal city regex fallback
        self.coastal_cities = [
            'mumbai','chennai','kolkata','kochi','goa','visakhapatnam','mangalore','puducherry','thiruvananthapuram','bhubaneswar'
            'surat','vadodara','rajkot','jamnagar','dwarka','somnath','puri','paradip','haldia','kakinada','machilipatnam','nellor
        ]
        print("✅ AI Engine initialized successfully!")

    # Language detection & translation
    def detect_language(self, text: str) -> str:
        try: return detect(text)
        except LangDetectError: return 'en'

    def translate_text(self, text: str, target_lang='en') -> Dict:
        try:
            src_lang = self.detect_language(text)
            if src_lang == target_lang: return {'original': text,'translated': text,'source_language': src_lang}
            translated = self.translator.translate(text, dest=target_lang)
            return {'original': text,'translated': translated.text,'source_language': src_lang}
        except: return {'original': text,'translated': text,'source_language':'unknown'}

    # Location extraction
    def extract_location(self, text: str) -> str:
        doc = nlp(text)
        for ent in doc.ents:
            if ent.label_ == "GPE": return ent.text
        text_lower = text.lower()
        for city in self.coastal_cities:
            if city in text_lower: return city.title()
        return "Unknown"

    # Urgency calculation
    def calculate_urgency(self, text: str, sentiment_scores: List) -> str:
```

```
        text_lower = text.lower()
        score = sum(3 for w in self.urgency_keywords['high'] if w in text_lower)
        score += sum(2 for w in self.urgency_keywords['medium'] if w in text_lower)
        score -= sum(1 for w in self.urgency_keywords['low'] if w in text_lower)
        negative_score = next((s['score'] for s in sentiment_scores[0] if s['label']=='LABEL_0'),0)
        if score >= 3 or negative_score>0.7: return "HIGH"
        elif score>=1 or negative_score>0.5: return "MEDIUM"
        else: return "LOW"

    # Analyze single post
    def analyze_post(self, text: str, location: str = None) -> Dict:
        tr = self.translate_text(text)
        txt = tr['translated']
        sentiment_results = self.sentiment_analyzer(txt)
        top_sentiment = max(sentiment_results[0], key=lambda x:x['score'])
        hazard_result = self.classifier(txt, self.hazard_labels)
        if location is None: location = self.extract_location(text)
        urgency = self.calculate_urgency(txt, sentiment_results)
        return {
            'timestamp': datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            'original_text': text,
            'translated_text': txt if tr['source_language']!='en' else None,
            'language': tr['source_language'],
            'location': location,
            'hazard_type': hazard_result['labels'][0],
            'hazard_confidence': round(hazard_result['scores'][0],3),
            'sentiment': top_sentiment['label'],
            'sentiment_score': round(top_sentiment['score'],3),
            'urgency_level': urgency,
            'all_hazard_scores': {l:round(s,3) for l,s in zip(hazard_result['labels'],hazard_result['scores'])}
        }

    # Batch analysis
    def batch_analyze(self, posts: List[Dict]) -> List[Dict]:
        results=[]
        for post in posts:
            text = post.get('text','')
            loc = post.get('location',None)
            if text.strip(): results.append(self.analyze_post(text, loc))
        return results

    # Hotspot generation
    def generate_hotspots(self, analysis_results: List[Dict], min_reports: int = 2) -> List[Dict]:
```

```
        loc_data = {}
        for res in analysis_results:
            loc = res['location']
            if loc=="Unknown": continue
            if loc not in loc_data:
                loc_data[loc] = {'total_reports':0,'high_urgency':0,'medium_urgency':0,'hazard_types':{},'avg_confidence':0,'lates
            d = loc_data[loc]
            d['total_reports']+=1
            if res['urgency_level']=='HIGH': d['high_urgency']+=1
            elif res['urgency_level']=='MEDIUM': d['medium_urgency']+=1
            h = res['hazard_type']
            d['hazard_types'][h] = d['hazard_types'].get(h,0)+1
            d['avg_confidence']+=res['hazard_confidence']
            if res['timestamp']>d['latest_report']: d['latest_report']=res['timestamp']
        hotspots=[]
        for loc,d in loc_data.items():
            if d['total_reports']>=min_reports:
                d['avg_confidence']/=d['total_reports']
                if d['high_urgency']>0: urgency='HIGH'
                elif d['medium_urgency']>0: urgency='MEDIUM'
                else: urgency='LOW'
                primary_hazard=max(d['hazard_types'].items(),key=lambda x:x[1])
                hotspots.append({
                    'location':loc,'total_reports':d['total_reports'],'urgency_level':urgency,
                    'primary_hazard':primary_hazard[0],'hazard_count':primary_hazard[1],
                    'avg_confidence':round(d['avg_confidence'],3),
                    'high_urgency_reports':d['high_urgency'],
                    'medium_urgency_reports':d['medium_urgency'],
                    'latest_report':d['latest_report'],
                    'all_hazards':d['hazard_types']
                })
        return sorted(hotspots,key=lambda x:( {'HIGH':3,'MEDIUM':2,'LOW':1}[x['urgency_level']],x['total_reports']),reverse=True)
```

```
# Step 3: Test posts
ai_engine = OceanHazardAI()

test_posts = [
    {'text': "Massive tsunami waves hitting Marina Beach Chennai! People running! #emergency #tsunami"},
    {'text': "मुंबई में भारी बारिश, सड़कों पर पानी भर गया है। सभी सावधान रहें!"},
    {'text': "Beautiful sunset at Goa beach today. Weather is perfect for swimming!"},
    {'text': "ஆபத்து! Chennai harbor near high tide. Warning issued!"},
    {'text': "Sunny day at Goa beach, tourists enjoying."},
```

```
    {'text': "Massive flooding in Mumbai streets after heavy rainfall. Rescue teams deployed."},
]

results = ai_engine.batch_analyze(test_posts)
```

```
🤖 Initializing Ocean Hazard AI Engine...
📝 Loading sentiment analysis model...
/usr/local/lib/python3.12/dist-packages/huggingface_hub/file_download.py:945: FutureWarning: `resume_download` is deprecated and wi
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                                929/929 [00:00<00:00, 81.0kB/s]
/usr/local/lib/python3.12/dist-packages/transformers/utils/generic.py:311: FutureWarning: `torch.utils._pytree._register_pytree_nod
  torch.utils._pytree._register_pytree_node(
pytorch_model.bin: 100%                                          501M/501M [00:06<00:00, 122MB/s]
Some weights of the model checkpoint at cardiffnlp/twitter-roberta-base-sentiment-latest were not used when initializing RobertaFor
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model trained on another task
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model that you expect to b
vocab.json:       899k/? [00:00<00:00, 2.06MB/s]
merges.txt:       456k/? [00:00<00:00, 3.43MB/s]
special_tokens_map.json: 100%                                    239/239 [00:00<00:00, 5.39kB/s]
🎯 Loading hazard classification model...
/usr/local/lib/python3.12/dist-packages/transformers/pipelines/text_classification.py:105: UserWarning: `return_all_scores` is now
  warnings.warn(
config.json:      1.39k/? [00:00<00:00, 21.7kB/s]
pytorch_model.bin: 100%                                          890M/890M [00:17<00:00, 111MB/s]
tokenizer_config.json: 100%                                      26.0/26.0 [00:00<00:00, 1.14kB/s]
vocab.json:       899k/? [00:00<00:00, 23.8MB/s]
merges.txt:       456k/? [00:00<00:00, 12.9MB/s]
special_tokens_map.json: 100%                                    772/772 [00:00<00:00, 40.0kB/s]
🌍 Loading translation service...
✅ AI Engine initialized successfully!
```

```
# Step 4: Print results
for i,res in enumerate(results):
```

```
    print(f"\n--- Post {i+1} ---\n{res}")
```

```
--- Post 1 ---
{'timestamp': '2025-09-24 17:14:24', 'original_text': 'Massive tsunami waves hitting Marina Beach Chennai! People running! #emergen

--- Post 2 ---
{'timestamp': '2025-09-24 17:14:28', 'original_text': 'मुंबई में भारी बारिश, सड़कों पर पानी भर गया है। सभी सावधान रहें!', 'translated_text': 'Heavy r

--- Post 3 ---
{'timestamp': '2025-09-24 17:14:30', 'original_text': 'Beautiful sunset at Goa beach today. Weather is perfect for swimming!', 'tra

--- Post 4 ---
{'timestamp': '2025-09-24 17:14:34', 'original_text': 'ஆபத்து! Chennai harbor near high tide. Warning issued!', 'translated_text': N

--- Post 5 ---
{'timestamp': '2025-09-24 17:14:37', 'original_text': 'Sunny day at Goa beach, tourists enjoying.', 'translated_text': None, 'langua

--- Post 6 ---
{'timestamp': '2025-09-24 17:14:40', 'original_text': 'Massive flooding in Mumbai streets after heavy rainfall. Rescue teams deploy
```

```
import pandas as pd

# Convert results into DataFrame
df = pd.DataFrame(results)
df_display = df[['original_text','location','hazard_type','hazard_confidence','sentiment','urgency_level']]

# Apply styling
def highlight_urgency(val):
    if val == 'HIGH':
        return 'background-color: red; color: white; font-weight: bold'
    elif val == 'MEDIUM':
        return 'background-color: orange; color: black; font-weight: bold'
    else:
        return 'background-color: green; color: white'

def bold_text(val):
    return 'font-weight: bold'

styled_df = df_display.style.applymap(highlight_urgency, subset=['urgency_level'])\
                            .applymap(bold_text, subset=['hazard_type','sentiment'])\
                            .set_properties(**{'text-align': 'left'})\
                            .set_table_styles([{
```

```
                                'selector': 'th',
                                'props': [('background-color', '#40466e'),
                                          ('color', 'white'),
                                          ('font-size', '14px'),
                                          ('text-align', 'center')]
                            }])

styled_df
```

```
/tmp/ipython-input-859250521.py:19: FutureWarning: Styler.applymap has been deprecated. Use Styler.map instead.
  styled_df = df_display.style.applymap(highlight_urgency, subset=['urgency_level'])\
/tmp/ipython-input-859250521.py:20: FutureWarning: Styler.applymap has been deprecated. Use Styler.map instead.
  .applymap(bold_text, subset=['hazard_type','sentiment'])\
```

	original_text	location	hazard_type	hazard_confidence	sentiment	urgency_level
0	Massive tsunami waves hitting Marina Beach Chennai! People running! #emergency #tsunami	Chennai	**high waves**	0.674000	**negative**	HIGH
1	मुंबई में भारी बारिश, सड़कों पर पानी भर गया है। सभी सावधान रहें!	Unknown	**flooding**	0.945000	**negative**	LOW
2	Beautiful sunset at Goa beach today. Weather is perfect for swimming!	Goa beach	**normal weather**	0.241000	**positive**	LOW
3	ஆபத்து! Chennai harbor near high tide. Warning issued!	Chennai	**dangerous currents**	0.273000	**neutral**	HIGH
4	Sunny day at Goa beach, tourists enjoying.	Goa beach	**normal weather**	0.402000	**positive**	LOW
5	Massive flooding in Mumbai streets after heavy rainfall. Rescue teams deployed	Mumbai	**flooding**	0.950000	**negative**	HIGH

```
import plotly.express as px

fig = px.data.tips()  # Dummy example; replace with your df_display
import plotly.graph_objects as go

fig = go.Figure(data=[go.Table(
    header=dict(values=list(df_display.columns),
                fill_color='darkblue',
                font=dict(color='white', size=14),
                align='left'),
    cells=dict(values=[df_display[col] for col in df_display.columns],
               fill_color=[['lightcoral' if lvl=='HIGH' else 'lightyellow' if lvl=='MEDIUM' else 'lightgreen'
                            for lvl in df_display['urgency_level']] for _ in df_display.columns],
```

```
                align='left'))
])

fig.show()
```

original_text	location	hazard_type	hazard_confidence	sentiment	urgency_level
Massive tsunami waves hitting Marina Beach Chennai! People running! #emergency #tsunami	Chennai	high waves	0.674	negative	HIGH
मुंबई में भारी बारिश, सड़कों पर पानी भर गया है। सभी सावधान रहें!	Unknown	flooding	0.945	negative	LOW
Beautiful sunset at Goa beach today. Weather is perfect for swimming!	Goa beach	normal weather	0.241	positive	LOW
ஆபத்து! Chennai harbor near high tide. Warning issued!	Chennai	dangerous currents	0.273	neutral	HIGH
Sunny day at Goa	Goa beach	normal weather	0.402	positive	LOW

```
# Step 5: Generate hotspots
hotspots = ai_engine.generate_hotspots(results)
print(f"\n🚨 Hotspots found: {len(hotspots)}")
for h in hotspots: print(h)
```

```
🚨 Hotspots found: 2
{'location': 'Chennai', 'total_reports': 2, 'urgency_level': 'HIGH', 'primary_hazard': 'high waves', 'hazard_count': 1, 'avg_confid
```

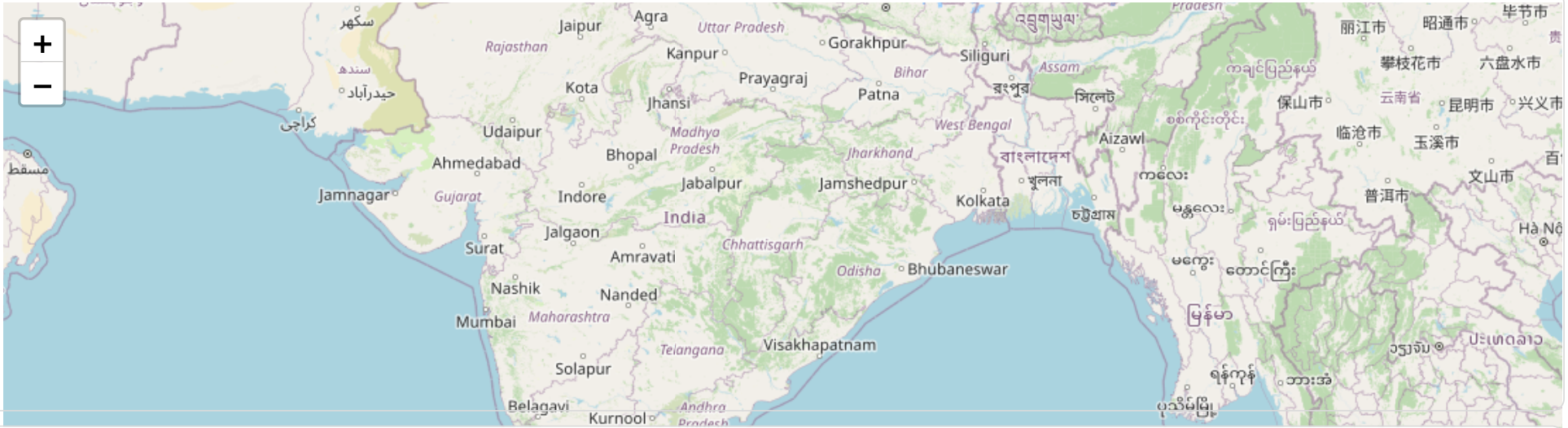
```
{'location': 'Goa beach', 'total_reports': 2, 'urgency_level': 'LOW', 'primary_hazard': 'normal weather', 'hazard_count': 2, 'avg_c
```

```
# Step 6: Optional map visualization
import folium

city_coords = {"Chennai":[13.0827,80.2707],"Mumbai":[19.0760,72.8777],"Goa":[15.2993,74.1240]}
m = folium.Map(location=[20,77], zoom_start=5)

for h in hotspots:
    loc = h['location']
    if loc in city_coords:
        folium.CircleMarker(
            location=city_coords[loc],
            radius=5 + h['avg_confidence']*10,
            popup=f"{loc}\nHazard: {h['primary_hazard']}\nUrgency: {h['urgency_level']}",
            color='red' if h['urgency_level']=='HIGH' else ('orange' if h['urgency_level']=='MEDIUM' else 'green'),
            fill=True
        ).add_to(m)

m
```




```
import matplotlib.pyplot as plt
from collections import Counter

hazards = [res['hazard_type'] for res in results]
counts = Counter(hazards)
plt.bar(counts.keys(), counts.values(), color='skyblue')
plt.title("Hazard Types Detected")
plt.ylabel("Number of Posts")
plt.xticks(rotation=45)
plt.show()
```