Great! Here's a **complete working code** for a **Fake News Detection System using NLI (Natural Language Inference)** with transformers by Hugging Face. This is a single Python script that uses the facebook/bart-large-mnli model to reason whether a given news statement contradicts a trusted fact.

**✅ Requirements**

Before running the code, install the required libraries:

pip install transformers torch streamlit

**🧾 Full Code: fake\_news\_nli.py**

import streamlit as st

from transformers import AutoTokenizer, AutoModelForSequenceClassification, pipeline

# Load the pretrained Natural Language Inference model

model\_name = "facebook/bart-large-mnli"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForSequenceClassification.from\_pretrained(model\_name)

# Create an inference pipeline

nli\_pipeline = pipeline("text-classification", model=model, tokenizer=tokenizer)

# Fake news detection function

def detect\_fake\_news(statement, trusted\_fact):

input\_text = f"{statement} </s></s> {trusted\_fact}"

result = nli\_pipeline(input\_text)[0]

return result["label"], result["score"]

# Simple decision logic

def is\_fake\_news(label, score, threshold=0.8):

return label == "CONTRADICTION" and score > threshold

# Streamlit UI

st.title("📰 Fake News Detector using Natural Language Reasoning")

st.write("Enter a news statement and a trusted fact to check if the statement is potentially fake.")

statement = st.text\_area("🗞️ News Statement", height=100)

trusted\_fact = st.text\_area("📚 Trusted Fact or Reference", height=100)

if st.button("🧠 Analyze"):

if statement and trusted\_fact:

label, score = detect\_fake\_news(statement, trusted\_fact)

st.write(f"\*\*NLI Result:\*\* {label} (Confidence: {score:.2f})")

if is\_fake\_news(label, score):

st.error("🚨 This may be Fake News (Contradicts trusted fact)")

elif label == "ENTAILMENT":

st.success("✅ Likely True (Entails trusted fact)")

else:

st.warning("🤔 Uncertain or Neutral — needs further verification")

else:

st.warning("Please fill in both fields.")

**▶️ How to Run**

Save the code as fake\_news\_nli.py and run:

streamlit run fake\_news\_nli.py

**🧪 Example Inputs to Try**

**News Statement:**

"The COVID-19 vaccine contains microchips."

**Trusted Fact:**

"There is no scientific evidence that COVID-19 vaccines contain microchips."

Would you like me to generate a **Google Colab notebook** version for quick testing online?

Great question! Understanding the **difference between using NLP and NLR** in projects is important because they serve **different purposes** and operate at **different levels of language understanding**.

**⚔️ NLP vs NLR in Projects: Key Differences**

| **Feature** | **NLP (Natural Language Processing)** | **NLR (Natural Language Reasoning)** |
| --- | --- | --- |
| 🔍 **Goal** | Understand, process, and manipulate human language | Apply logical reasoning and inference on language |
| 🧠 **Depth of Understanding** | Surface-level understanding (tokenizing, parsing, classification) | Deeper understanding (logical entailment, cause-effect, commonsense) |
| ⚙️ **Techniques Used** | Tokenization, POS tagging, NER, text classification, summarization, translation, etc. | Natural Language Inference (NLI), logic-based reasoning, question answering, fact checking |
| 📈 **Complexity** | Lower to moderate | High (requires contextual + logical understanding) |
| 🤖 **Examples of Models** | BERT, GPT, spaCy, NLTK, T5 | BART-MNLI, RoBERTa-MNLI, GPT-4 (in reasoning mode), DeBERTa-NLI |
| 📦 **Common Tasks** | Sentiment analysis, topic modeling, chatbot intent detection, summarization | Fake news detection, fact verification, commonsense reasoning, entailment classification |
| 🧪 **Example Use Case** | Classify reviews as positive/negative | Check if a news statement contradicts a known fact |
| 🧰 **Data Used** | Raw text datasets | Premise-hypothesis pairs, knowledge bases, logic chains |

**🧠 Example**

Let’s compare how each approach might handle the task of **fake news detection**.

**🔹 Using NLP:**

* Train a model to classify news headlines as "fake" or "real" based on past labeled data.
* Might learn surface patterns ("clickbait = fake").

**🔹 Using NLR:**

* Take a news claim and **compare it to a verified fact**.
* Use **Natural Language Inference (NLI)** to reason:
  + Does the claim contradict the fact?
  + Does it entail or support the fact?
  + Is it neutral?

**🛠️ When to Use What?**

| **Project Goal** | **Use NLP when…** | **Use NLR when…** |
| --- | --- | --- |
| Chatbot intent | You want to extract intents from queries | You want the bot to reason with context and logic |
| Sentiment analysis | Simple opinion classification is enough | You want to explain *why* a sentiment is true |
| Fake news detection | You have labeled data | You want fact-checking against verified sources |
| Question answering | You can pull answers from a document | You want the model to **reason** over multiple facts |
| Document summarization | You need extractive or abstractive summary | You want cause-effect or logic-aware summaries |

**🚀 Summary**

| NLP = Understand + Process language |  
| NLR = Understand + Reason over language |

If your project needs **classification or structure extraction**, use **NLP**.  
If it needs **logic, verification, or commonsense**, go with **NLR**.

Would you like a practical comparison project (e.g., fake news detection using NLP **vs.** NLR side-by-side)?

[**https://github.com/RedFibre/Fake-News-Detector?utm\_source=chatgpt.com**](https://github.com/RedFibre/Fake-News-Detector?utm_source=chatgpt.com)

**I found the GitHub repository for a similar Chrome extension called Fake-News-Detector, developed by RedFibre. While not exactly the Reddit-posted "Deepfake News Detector," it follows a comparable concept—using AI models to flag potentially fake news. You can explore the source code here:**

* **GitHub: RedFibre/Fake-News-Detector – A Chrome extension with an accompanying Flask backend for text input and fake‑news prediction (**[**github.com**](https://github.com/RedFibre/Fake-News-Detector?utm_source=chatgpt.com)**).**

**Though the Reddit post mentions the extension being powered by BART‑large‑MNLI and Google Gemini, that exact GitHub link wasn’t publicly shared. Still, the RedFibre project can serve as a strong starting point if you want to build or adapt your own version.**

**🛠️ What You Can Do Next**

**1. Inspect the GitHub code  
Review the frontend (Chrome extension) and backend (Flask + ML model) to understand the architecture.**

**2. Integrate BART‑large‑MNLI  
Set up the facebook/bart-large-mnli from Hugging Face in the backend to perform zero-shot classification on article text.**

**3. (Optional) Add Gemini for Fact-Checking  
You can layer Gemini API calls for deeper fact verification after the initial "fake/real" inference.**

**Would you like help setting up the BART‑MNLI classifier in this project or tips on integrating Gemini?**

Seach engine id : 734837c73550d4dba

Api key : AIzaSyBryMskx\_DgR5kuitDJk60nhNlsRYQNk9k

Google custom search json api