FakeCheck Search Prototype: Search Engine for Verifying Online Claims

Hitarth Chopra
Texas A&M University
hitarth@tamu.edu
NetID: hitarth

David Zhao
Texas A&M University
sixingzhao@tamu.edu
NetID: sixingzhao

Tony Vu
Texas A&M University
quocduyvu6262@tamu.edu
NetID: quocduyvu6262

1 What is the function of your tool?

FakeCheck Search is a prototype search engine that helps users verify factual claims. When a user inputs a statement (e.g., "Electric cars cause more pollution than gasoline cars"), the system retrieves relevant evidence from reliable news or web sources, classifies each snippet as *supporting*, *refuting*, or *neutral* toward the claim, and presents a concise summary verdict with clear visual indicators and source links. Unlike a generic search engine, it emphasizes evidence polarity and transparency.

2 Why do we need such a tool and who will use it?

Misinformation and cherry-picked evidence are widespread online. Most users lack the time or skill to manually verify multiple sources. This tool benefits:

- General users who want to quickly check claims from social media (e.g., Reddit, X, TikTok).
- Journalists and students verifying statements for reporting or research.
- Educators studying public information credibility.

It makes search results more trustworthy and interpretable, promoting media literacy.

3 Does this kind of tool already exist? How is yours different?

Yes—but only partially. Existing tools such as Google Fact Check Explorer search known fact-check articles, while ClaimBuster detects factual claims but does not verify them. Human-written sites like Snopes and Full Fact provide manual fact checks for limited claims.

Our system differs by:

- Handling arbitrary user claims, not just known ones.
- Showing evidence polarity (support/refute/neutral) transparently.
- Allowing interactive exploration instead of static verdicts.

People will care about the difference because they can instantly see both sides of evidence rather than a single "true/false" label. The main challenge lies in balancing accuracy, explainability, and efficiency—especially when retrieving and classifying noisy, contradictory web information.

4 How do you plan to build it?

1. **Claim Embedding:** Convert user claim to a semantic vector using SentenceTransformer (`all-MiniLM-L6-v2`).
2. **Evidence Retrieval:** Use FAISS or ElasticSearch to retrieve top-k related passages from Wikipedia or a NewsAPI dataset.
3. **Entailment Classification:** Use a Natural Language Inference model (`roberta-large-mnli`) to label retrieved evidence as support/refute/neutral.
4. **Summarization and Ranking:** Aggregate evidence and optionally summarize using a small `T5-small` model.
5. **Frontend:** Implement an interactive interface using Streamlit or Gradio for visualization.

5 What existing resources will you use?

We will use pretrained models from HuggingFace (SentenceTransformers, NLI models, T5), datasets such as Wikipedia dumps, PolitiFact, and NewsAPI.org articles, and libraries like FAISS, Transformers, and Streamlit. For evaluation, we will use the FEVER dataset for claim–evidence correctness.

6 How will you demonstrate usefulness?

- **Live demo:** User enters claim → retrieved evidence shown with polarity colors.
- **Case study:** Compare with Google search results to show improved interpretability.
- **Evaluation:** Measure entailment classification accuracy on a labeled set.
- **User feedback:** Short survey on trust and clarity of results.

7 Rough timeline and milestones

Week 1	Define scope, collect datasets (Wikipedia, news)
Week 2	Implement embedding + retrieval pipeline
Week 3	Integrate entailment classification model
Week 4	Build UI with evidence polarity visualization
Week 5	Evaluate and refine models
Week 6	Record demo and finalize report