DATS 598: Data Science Capstone

Project Proposal

A Statistical Truth Detection System

Eric Crisp
University of Pennsylvania
ecrisp@upenn.edu

Thursday 11th September, 2025

Contents

A Statistical Truth Detection System

Date: Thursday 11th September, 2025

Motivation and Significance

This will be an end-to-end implementation of a fact-checking NLP application. The premise behind this project is to develop a process that can be used to determine how likely a statement contains the truth. This is motivated by the common incorrect responses, or general mistrust, in LLM outputs. While this tool is not designed to extend to all modes of truth detection such as mathematics, the extension would track similarly. At the end of the project, the goal is to have a simple web app that allows the user to select one of several LLM API's, write a prompt and query the LLM, have the output be fed into the application and be used as input to the truth detection process. Ideally, the output would be some notion of confidence such as a percentage (with, or without a confidence interval) rather than a binary ”true” or ”false”. The result should be a degree of confidence given to the LLM generated text by a third party (the external application) on the LLM generated output.

Project Plan and Timeline

As mentioned above, this project is motivated by the general mistrust in LLM outputs for simple things such as the AI-assisted results when Googling something like ”Did we land on the moon?”. LLMs include things like RLHF, Constitutional AI, and being trained on so many perspectives and opinions that inherent bias is reduced. This application is not intended to be in replacement for any training process. Where possible, this process is an attempt at illustrating how factual various LLM responses can be over a relatively narrow scope, or dataset.

The process should behave similar to a normal LLM query. For example, the user would prompt the application with something like ”There are many movies that made $1T in revenue.” Clearly, this is wrong. This prompt would be analyzed with various NLP processes using open source libraries to understand the point of the question at hand. The prompt would simultaneously be fed to an LLM of that the user selected and the output is then processed by the fact-checking application. Things like sentence embedding, similarity classification, entity recognition, and fact extraction. Using trained models such as BERT that have been fine tuned with FEVER dataset could be used in conjunction for truth-detection. After NLP processing of both the query and the LLM generated response, a degree of confidence would be calculated on the LLM output. As the final result, the LLM output would be returned along with some quantification of the likelihood the response is true. If the LLM returns ”Yes, several movies have earned $1T in revenue” the coupled truth-detection result should be nearly 0%.

For determining ”truth”, fine tuning larger models on specifically the FEVER, Factuality, or Wikidata datasets is likely one avenue for gaining confidence in the LLM response. However, there are other methods that could be implemented such as hypothesis testing, or even RAG methods. Libraries for many of these processes exist including HuggingFace, scikit-learn, Haystack, and LangChain, among many other open source resources.

There are several obstacles involved with this process. Some of which are merely learning as my current exposure to NLP, fine tuning large models, and building web apps is limited, however, this pipeline and technical scope seems doable. The timeline and milestones are described below. I aim to gain experience creating an end-to-end solution that leverages open source libraries while creating tools and supporting functionality where necessary. I anticipate problems mostly in the fact finding portion of this project. Determining what part of the prompt is the ”fact” that should be checked for truth can be confusing and ambiguous.

The approach to generating the truth detection system is as follows:

Week 1: Setup & Data Pipeline

- Set up development environment with Python, virtual environment, and dependencies
- Download and explore FEVER, Factily, Wikidata dataset(s)
- Build data preprocessing pipeline
- Create basic data visualization and statistics
- **Deliverables:**
  - ☐ Working Python environment
  - ☐ FEVER dataset loaded and preprocessed
  - ☐ Basic statistics dashboard showing dataset distribution
  - ☐ Data cleaning and preparation functions

Week 2: Create Initial Model

- Implement sentence embedding baseline using SentenceTransformers
- Build similarity-based classification
- Integrate Wikipedia API for evidence retrieval
- Create model evaluation framework
- **Deliverables:**
  - ☐ Baseline fact-checker achieving on FEVER validation set
  - ☐ Wikipedia API integration for real-time evidence retrieval
  - ☐ Evaluation metrics (accuracy, F1, precision/recall)
  - ☐ Create evidence ranking and filtering system

Week 3: LLM Integration & NLP on Query

- Integrate OpenAI API for AI responses
- Build claim extraction system using NLP techniques via open source libraries (from scratch as needed)
- Implement factual claim detection and entity extraction
- Create end-to-end pipeline: Query → LLM output → Claims → Truth Detection
- **Deliverables:**
  - ☐ AI query handling process (limited token on input and output)
  - ☐ Extract claim and facts from query
  - ☐ Named entity recognition for people, places, dates, numbers via NLP processes

- ☐ Working pipeline prototype

Week 4: Advanced Verification Model

- Incorporate BERT/RoBERTa-based fact verification model via fine tuning on FEVER, or other dataset (Wikidata)
- Wrap up the statistical truth detection process (confidence internals, hypothesis testing, A/B testing, weighted averages of results, etc.)
- **Deliverables:**
  - ☐ Fine-tuned transformer model for fact verification
  - ☐ Confidence scoring system with multiple methods
  - ☐ Performance benchmarking against FEVER leaderboard

Week 5: Begin Web App Framework

- Build web application interface
- Create user input methods and result displays
- Add basic error handling and user feedback
- **Deliverables:**
  - ☐ Functional web interface for asking questions
  - ☐ Basic responsive design

Week 6: Continued Web App Development

- Add result visualization (hooks for statistical result and LLM output)
- Implement claim highlighting, or a summary of what was determined to be the ”fact”
- Create evidence source display with links
- Build confidence visualization (score, coloring from red to yellow to green depending on percentage true)
- **Deliverables:**
  - ☐ Annotation of LLM output
  - ☐ Truth detection output

Week 7: Begin Backend Development

- Convert core functionality to FastAPI backend
- Implement REST API endpoints
- Add request/response validation
- Build async processing for better performance (similar to 550 web app)
- **Deliverables:**
  - ☐ FastAPI backend with documented endpoints
  - ☐ Async pipeline verification
  - ☐ API rate limiting and authentication (OpenAI, Wikidata, etc.)

Week 8: Database Integration

- Use FEVER or another ground truth dataset in an SQLite or PostgreSQL database (or Amazon RDS)
- Implement SQLite/PostgreSQL for storing results
- Add user-based functionality (login, result storage, etc.)
- **Deliverables:**
  - ☐ Database schema for FEVER access (if needed)
  - ☐ Query history and user data

Week 9: Multi-Source Integration

- Integrate multiple free sources (e.g., Wikidata, government APIs)
- Explore integrating Retrieval-Augmented Generation (RAG) for real-time or recent facts (e.g., ”The S&P fell 3 points today”)
- **Deliverables:**
  - ☐ Source credibility database and scoring algorithm
  - ☐ Integration with Wikidata for structured facts
  - ☐ Multi-source evidence aggregation
  - ☐ Time-sensitive claim handling

Week 10: Deployment & Monitoring

- Deploy/host web app to cloud platform
- **Deliverables:**
  - ☐ Push production deployment

Week 11: Finishing Touches, Document, and Present

- Final UI/UX improvements
- Create (or clean up) documentation
- **Deliverables:**
  - ☐ Complete user documentation and help system
  - ☐ Presentation of process, results, and tooling

Technical Stack

- **Backend:**
  - Python
  - Either Flask or FastAPI for API endpoints
  - SQLite/PostgreSQL for data storage
- **Machine Learning:**
  - HuggingFace Transformers, PyTorch, scikit-learn

- **Frontend:**
  - HTML/CSS/JavaScript
  - Plotly for visualizations
- **External APIs:**
  - OpenAI, Wikipedia, Google Fact Check API
- **Deployment:**
  - Docker containers
  - GitHub for CI/CD

Data Sources

All data being used is open source. FEVER, Factily, Wikidata, and any other data sources are (currently) being considered as ”true” - verified and trusted sources.

- **FEVER** – A large-scale dataset for fact-checking based on Wikipedia.
  https://fever.ai
- **Factify** – A dataset and benchmark for factuality detection.
  https://github.com/surya1701/Factify-2.0
- **Wikidata** – A collaboratively edited knowledge base from the Wikimedia Foundation.
  https://www.wikidata.org
- **BERT** – A language model developed by Google AI.
  https://arxiv.org/abs/1810.04805
- **T5** – A unified framework that casts all NLP tasks as text generation problems.
  https://arxiv.org/abs/1910.10683

Mentor Credentials

- **Name:** Jelle Vanhorenbeke
- **Email:** jellevh@seas.upenn.edu
- **Phone Number:** (XXX) XXX-XXXX
- **Relevant Skills and Experience:** Anthropic, Member of the Technical Staff - Software Engineer in Research (several years of experience in development)
- **LinkedIn:** https://www.linkedin.com/in/jellevanhorenbeke/