NoCap: Fact Checking with AI

Anthony Ciero, Joshua Pechan, Varun Doddapaneni, Thomas Chamberlain
Faculty Advisor: Professor Silaghi

Goals/Motivation

- Determining an article to be fact or fiction should be easy and comparative.
- After determining validity, it should be common knowledge and it should be spread to everyone.
- The goal is to create a website that checks articles, specifies what's fact and fiction, and gives it a rating.
- We then aggregate scores of articles by the same publishers to give them an "authenticity" rating and allow users to view their scores and the articles'.

Approach

- **AI Powered Claim Detection**
  - User insert article URL
  - AI gives article a rating based on authenticity
  - If article has not been used before, store in database
- **Publisher Authenticity**
  - Aggregate authenticity scores from the same publisher
  - Give publishers an average authenticity score based on articles in database
  - Allows users to browse by publisher, view their score, and database articles' scores
- **Chrome Extension**
  - Use webpage features on the article's link
  - Run extension to grab link and perform the usual features

Novel Features

- **Authenticity Graphical Representation**
  - Shows graph with misinformation rating
  - Shows which words are most shown up in true vs false articles
- **Web Content Accessibility Guidelines (WCAG)**
  - Ensure website follows accessibility standards
  - Being able to tab between important sections of the website
  - Accessible colors for background/text combinations

Algorithms/Tools

- Python (backend): primary server-side language for AI orchestration and services.
- FastAPI (API/backend web framework): lightweight, async-friendly framework to expose REST endpoints.
- LangChain & LangGraph (LLM/NLP modules): tooling to compose prompts, retrieval, and multi-step AI workflows.
- React (JavaScript UI): component-based interface for the extension popup and web dashboard.
- AWS Bedrock (Nova Lite): managed LLMs with model swapability for classification and analysis tasks.
- AWS Amplify (GraphQL with AppSync + DynamoDB): optional persistence layer for user preferences, cached verdicts, and analytics.

Technical Challenges

- **Frontend Design**
  - More familiar with backend development
  - Must ensure frontend is designed simply and accessible for all
- **LangChain/LangGraph**
  - Our familiarity with AI is not all at the same level
  - Important for managing prompts/reasoning chains in large language models
  - Start with smaller prototypes with simple prompts before going full-scale
- **AI Connection**
  - Must ensure proper procedure of managing API calls and handling authentication securely
- **Chrome Extension**
  - Create simple and effective chrome extension to stand out from other web-based fact checkers

Design: 3 Main Pages

- **Home Page:** Default starting page when the website is opened, allowing the user to paste a desired article or block of text that they wish to fact check.
- **Report Page:** Page that comes up after a user enters their desired article or selects a stored article from the Database Page. It shows information about the article like title, author, publisher, and the URL. After, it shows the authenticity score of the article followed by a summary of why the AI rated the article this way and a detailed explanation of what it believed to be fact or fiction.
- **Database Page:** Where all authenticated articles are stored for users to search through. These articles are stored in publisher cards that show the publisher's name as well as aggregate authenticity score of articles. The articles are stored in their own cards showing authenticity score on the right side and article title, author, publication date, and URL. The authenticity scores are color-coded, with green, yellow, or red depending on score.

Evaluation

- Speed: ensure that article reports are created within 30 seconds
- Accuracy: Currently our model will give a wide range of authenticity scores for the same article. Our goal is to concise this down as much as possible.
- User Survey: Conduct user surveys on specific parts of our software like creating an article report, searching/filtering for an article in our database, (e.g. rating of 1-5 on each of the different features)

Progress Summary

Model/Feature	Completion %	To Do
Frontend	80%	Add example articles to home page, connect article data via input on Home Page
Database	75%	Connect to frontend article input to save in database
AI Model	100%	Connected to AI model allowing score and report generation
Prompt Engineering	75%	Improve prompt to yield better/more reliable score.
Chrome Extension	0%	One of main goals for semester 2

Milestone 4

- Improve prompt engineering
- Improve model output
- Show default home page cards
- Article data connection to report page via input
- Create logo and branding
- Chrome extension

Milestone 5

- Finalize prompt engineering
- Ensure consistent model output
- Create graphs/visualizations for collected data
- Conduct evaluation and analyze results
- Create poster for Senior Design Showcase

Milestone 6

- Finalize data graphs/visualizations
- Test/demo of the entire system
- Conduct evaluation and analyze results
- Create user/developer manual
- Create demo video

Milestone 4 Task Matrix

Task	Thomas	Anthony	Josh	Varun
1. Prompt Engineering	0%	0%	50%	50%
2. Improve model output	0%	0%	50%	50%
3. Show default home page cards	50%	50%	0%	0%
4. Article data connection to report page via input	30%	0%	0%	70%
5. Create logo and branding	50%	50%	0%	0%
6. Chrome extension	50%	50%	0%	0%