Final Tech Spec

[project name]

Kunal Srivastava, Akash Shetty, Pablo Ahmadi, Connor Chan

Problem Definition

Introduction

In the past year, 77% of health related information was found via general search engines like Google, Bing, or Yahoo. Only 13% of online health seekers referred to a site that specializes in health information. As the pandemic showed us, this a dangerously low number. Misinformation is a powerfully destructive force, especially as it relates to our health but also in

As an example, early during the pandemic, questions were raised regarding the possible use of disinfectants administered internally to patients with COVID-19. On June 5th, the CDC reported a steep increase in calls to poison centers regarding exposure to household disinfectants. A CDC survey found that 39% of responders engaged in dangerous practices including washing food products with bleach, applying household cleaners directly to skin, and intentionally inhaling or ingesting disinfectants with the goal of preventing COVID-19 infection. According to industry experts, rampant misinformation across the internet helped make this a reality.

But the issue isn't purely the lack of credible information, it's also their lack of use. To this day, credible alternatives such as PubMed and Google Scholar with thousands of reputable medical publications are grossly underutilized by the general public. Reasons for why include the general lack of awareness to their existence, the inability for the average individual to assess their credibility, and inexperience with navigating their vast, data-heavy platforms. The general public deserves a better way to receive credible health information on the internet.

Varro is a search engine that provides users with only research-backed information written by experts, all presented in a user-friendly and easily digestible format. By leveraging AI to translate complicated text into concise prompts, we can provide users with only the most relevant articles from a comprehensive research database, such as Semantic Scholar. Varro offers users access to relevant, research-grade articles presented in a clear, concise, and easily understandable manner. It provides a summary of each article, along with easily interpreted statistics to evaluate bias, the extent of academic support, and readability. Users are no longer subject to the rampant misinformation and heavily biased language that plagues general search engines, or the heavily technical, underutilized research databases. Varro makes finding credible information easily accessible to the general public.

Varro: an AI search engine that provides users with only research-backed information written by experts, all presented in an easily digestible format. It offers summaries and stats for bias, academic backing, and readability, thus combating misinformation and making credible info accessible.

Technical Proposal

We propose to build a website where users can paste long-form text from research papers into the website. On the backend, we then use OpenAI API to condense the long-form text into a short research-oriented, searchable query that could be pasted into a search engine like Semantic Scholars. Using the Semantic Scholars API, we will perform a search with the condensed query. With the information retrieved by the API, in addition to the OpenAI API and the given input, we will provide relevant information (described below) in a clear and engaging UI on the website.

TLDR

Short and concise summary of the input. This TLDR should be sure to maintain the emotion, bias, and viewpoints of the original input.

Contextual Bias

***of the input*

- Contextual Bias (OpenAI API)

    - Very important to clearly design and communicate how this is calculated to users

Contextual Information

***for each related article*

- Title (Semantic Scholars API)
- Authors (Semantic Scholars API)
- Journal (Semantic Scholars API)
- Num. of citations (Semantic Scholars API)
- Num. of references (Semantic Scholars API)
- Readability Difficulty Score (Flesch-Kincaid Grade Level formula)
- Sentiment Analysis (VADER: Valence Aware Dictionary and sEntiment Reasoner)

https://ojs.aaai.org/index.php/ICWSM/article/view/14550

https://www.scimagojr.com/journalrank.php?type=j

(H-Index ranking)

Tech Stack:

Backend

- Python + Flask + OpenAI API + Semantic Scholars API

Frontend

- React + Axios (HTTP requests from React to Flask server) + Material-UI (design)

Data Processing

- Python (Flesch-Kincaid Grade Level Formula, Valence Aware Dictionary and sEntiment Reasoner)

Deployment

- **Docker**: To containerize your application. This makes it easier to manage dependencies and deploy your application.
- **AWS Elastic Beanstalk** or **Heroku**: These platforms support both Docker and Python, and they can automatically scale the application based on traffic.

Version Control

- **Git & GitHub**: For version control and collaboration.

I currently loop through

each key has the following information:

Given a positive, negative, and neutral number in between 0 and 1, I want to use React to write a function that takes these values and outputs a rectangular component has a green-to-red gradient from left to right, indicating green for positive sentiment and red for negative sentiment.

Then, given the three values, I want to perform a calculation to find the weighted sentiment. The formula for a weighted sentiment is (0 * negative + 50 * neutral + 100 * positive) / 100. Then, I want to display a dot on the gradient visualization to indicate the current weighted sentiment. Finally, I want to return this entire component. Currently im using Material UI to do this. The idea is that i want to call this function with different inputs to get their respective component showing the sentiment on a visualiza

Minimal Result

- Title
- Journal

- R → G
- TLDR
- Amount of citations

Backend Python Script

```
import openai
import re
import os
import requests
import textstat
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# nltk.download('vader_lexicon')
os.environ['OPENAI_API_KEY'] = 'INSERT_HERE'

def condense_to_query(long_form_text):
    openai.api_key = os.environ['OPENAI_API_KEY']

    preprompt = 'Imagine you are an intelligent AI, tasked with

    response = openai.Completion.create(
        engine="gpt-3.5-turbo-instruct",
        prompt=preprompt+long_form_text,
        temperature=0.5,
        max_tokens=60
    )

    generated_text = response.choices[0].text.strip()
    query = re.sub(r'\W+', ' ', generated_text)
    return query
```

```
def search_semantic(query):
    base_url = "https://api.semanticscholar.org/graph/v1"
    params = {
        "query": query,  # The keyword to search for
        "offset": 0,  # The number of results to skip
        "limit": 10,  # The number of results to return
        # The fields to return for each paper
        "fields": "title,authors,citationCount,referenceCount,jo
    }

    response = requests.get(base_url + "/paper/search", params=p
    if response.status_code != 200:
        return None
    return response.json()


def process_semantic_results(data):
    papers = []
    if data['data']:
        for paper in data['data']:
            paper_dict = {
                'title': paper['title'] if paper['title'] else N
                'authors': ', '.join(author['name'] for author i
                'journal': paper['journal']['name'] if paper['jo
                'citationCount': paper['citationCount'] if paper
                'referenceCount': paper['referenceCount'] if pap
                'abstract': paper['abstract'] if paper['abstract
            }
            if paper_dict['abstract'] is not None:
                if 'abstract' in paper_dict and isinstance(paper
                    readability_score = textstat.flesch_kincaid_
                else:
                    readability_score = None
                sia = SentimentIntensityAnalyzer()
                sentiment_scores = sia.polarity_scores(paper_dic
                paper_dict['readability_score'] = readability_sc
                paper_dict['sentiment_scores'] = sentiment_score

            papers.append(paper_dict)
        return papers
    return None



def print_data(papers):
    for paper in papers:
        print("Title:", paper['title'] if 'title' in paper else
        print("Authors:", paper['authors'] if 'authors' in paper
        print("Journal:", paper['journal'] if 'journal' in paper
        print("Num. Citations:", paper['citationCount'] if 'cita
        print("Num. References:", paper['referenceCount'] if 're
        print("Readability Difficulty Score:", f"{paper['readabi
        print("Sentiment Scores:", paper['sentiment_scores'] if

        print('\n\n')




long_form_text = """
Believable proxies of human behavior can empower interactive app
"""
query = condense_to_query(long_form_text)
print('query:', query)
data = search_semantic(query)
if data:
    papers = process_semantic_results(data)
    print_data(papers)
else:
    print("error")
```

Prompts:

1. Machine Learning in Healthcare

2. Machine learning has emerged as a transformative force in the field of molecular biology, revolutionizing our ability to process and interpret vast biological datasets. It plays a pivotal role in various applications, such as drug discovery and design, where virtual screening and drug repurposing benefit from predictive models that rapidly identify potential drug candidates. Machine learning has also reshaped protein biology, enabling the prediction of complex protein structures and functions, particularly crucial in understanding their roles in biological processes. In genomics and sequence analysis, ML aids in genome annotation, variant calling, and gene expression analysis, facilitating the study of genetic diseases and intricate cellular mechanisms. Moreover, structural biology leverages machine learning to simulate molecular dynamics and predict interactions between proteins and ligands, underpinning drug development. From image analysis to epigenomics, personalized medicine, and phylogenetics, machine learning's interdisciplinary impact in molecular biology continues to drive advancements and unlock deeper insights into the biological world.

currently

```
curl -u these:nuts -X POST -H "Content-Type: application/json" -d
'{"long_form_text":"Machine Learning in Healthcare"}' http://localhost:5000
```

Clone the repo

cd into repo

pip install -r requirements.txt

run app.py (now the backend is running)

open a new terminal tab

curl -u these:nuts -X POST -H "Content-Type: application/json" -d '{"long_form_text":"Machine Learning in Healthcare"}' http://localhost:5000

the above command is user/pw then sends an api call to the local backend server

Error(Akash):

```
html lang=en>
<title>400 Bad Request</title>
<h1>Bad Request</h1>
<p>Failed to decode JSON object: Expecting property name enclosed in double quotes: line 1 column 2 (char 1)</p>
```

```
{results["sentiment"]["neg"] > .5 || results["sentiment"]["pos"
                 <Typography variant="h7" fontStyle={"italic"}>
                 ) : (
                   // Render this when neg and pos are not above
                  <Typography variant="h7" fontStyle={"italic"}
              )}
```

```
<Box sx={{ width: 300 }}>
                          <Slider
                               // defKey={`slider-${10}`}

                              defaultValue= {[results[key][prope

                              //getAriaValueText={valuetext}
                              step={0.1}
                              marks
```

```
                                min={0}
                                max={50}
                                valueLabelDisplay="auto"
                                //disabled
                              />
                              </Box>
```

```
sx={{
                                "&.Mui-disabled": { // Define st
                                  color: 'red', // Change the co
                                },
                              }}
```

welcome to varo

varo is an engine for informaton quality analysis

https://flask-production-cbf0.up.railway.app/