

**ARGUSURE: A GEMINI-BASED CHROME EXTENSION  
FOR COUNTERARGUMENT GENERATION**

**CARL JOVEN M. MARASIGAN**


**PRESENTED TO THE FACULTY OF THE  
INSTITUTE OF COMPUTER SCIENCE  
COLLEGE OF ARTS AND SCIENCES  
UNIVERSITY OF THE PHILIPPINES LOS BAÑOS  
IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE  
DEGREE OF**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**JUNE 2024**

**This study can be accessed:**

<b>By the general public</b>	
<b>Only after consultation with the author and SP adviser</b>	<b>YES</b>
<b>By those bound by a confidentiality agreement</b>	

**Signature of Student:**  \_\_\_\_\_

**Signature of Adviser:** \_\_\_\_\_

**Signature of Director:** \_\_\_\_\_

The Faculty of the Institute of  
Computer Science of the University of the  
Philippines Los Baños accepts the Special  
Problem entitled

**ARGUSURE: A GEMINI-BASED CHROME EXTENSION  
FOR COUNTERARGUMENT GENERATION**

**CARL JOVEN M. MARASIGAN**

In partial fulfillment of the requirements for the  
Degree Bachelor of Science in Computer Science

**ASSOC. PROF. CONCEPCION L. KHAN**  
Adviser

---

Date Signed

**DR. MARIA ART ANTONETTE D. CLARIÑO**  
Director, Institute of Computer Science

---

Date Signed

## **ACKNOWLEDGEMENT**

I would like to express my utmost gratitude to everyone who helped and supported me throughout the completion of this study. To my adviser, Prof. Concepcion Khan, for her kind consideration and guidance. To DOST-SEI, for the financial assistance they provided. To my friends, for the joy, laughter, and sense of belonging they made me experience. And above all, to my family, whose love, care, support, and encouragement made me feel that, regardless of whatever happens, I always have someone who has my back.

## TABLE OF CONTENTS

	<u>PAGE</u>
<b>TITLE PAGE</b>	<b>i</b>
<b>APPROVAL PAGE</b>	<b>ii</b>
<b>ACKNOWLEDGEMENT</b>	<b>iii</b>
<b>TABLE OF CONTENTS</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>vi</b>
<b>ABSTRACT</b>	<b>vii</b>
<b>INTRODUCTION</b>	<b>1</b>
Background of the Study	1
Statement of the Problem	2
Significance of the Study	3
Objectives of the Study	4
Scope and Limitations	4
Date and Place of the Study	5
<b>REVIEW OF RELATED LITERATURE</b>	<b>6</b>
Argument and Counterargument Generation using Generative AI technologies	6
Prompting	8
Related Software Applications	9
<b>METHODOLOGY</b>	<b>11</b>
Development Tools	11
Features	13
Chrome Extension	13
Chrome Extension's Web Application	14
Use Case	15
Counterargument Generation Flowchart	16
Database	17
Application Testing and Evaluation	18
<b>RESULTS AND DISCUSSION</b>	<b>19</b>
Development	19

Development Environment	19
Gemini API	19
Prompting	20
Chrome Extension and Web Application	23
System Usability Testing	33
Counterargument Quality Evaluation	35
<b>CONCLUSION AND FUTURE WORK</b>	<b>38</b>
Conclusion	38
Future Work	39
<b>LITERATURE CITED</b>	<b>41</b>

## LIST OF TABLES

<b><u>TABLE</u></b>		<b><u>PAGE</u></b>
1	System Usability Scale Scores	35
2	Counterargument Quality Evaluation Scores	36

## LIST OF FIGURES

<b><u>FIGURE</u></b>	<b><u>PAGE</u></b>
1 Use Case Diagram for the Counterargument Generator Application	16
2 Counterargument Generation Flowchart	17
3 Prompt to determine the claim type of the input	21
4 Prompt to determine if the input is an argument	21
5 Prompt to determine if the input is a claim	22
6 Prompts for Counterargument Generation	22
7 Sign up Page	23
8 Sign in Page	24
9 Sign in or Sign up using Google Account	24
10 Window Popup Landing Page	25
11 Chrome Extension's Popup Landing Page	25
12 Context Menu	26
13 Home page	26
14 Chrome Extension's Popup	27
15 Window Popup with Generated Counterarguments	28
16 Chrome Extension's Popup with Generated Counterarguments	28
17 Home page with Generated Counterarguments	29
18 Save to Modal	29
19 Topic List Page	30
20 Saved Counterarguments Page	31
21 Sample Page for Counterarguments Viewing (Liked Counterarguments Page)	31
22 Search Results Sample	32

23	Profile Page	32
24	Admin Page Dashboard	33
25	Admin Page Users Section	34
26	Prompt for LLM-based Automatic Evaluation	37



## **ABSTRACT**

**CARL JOVEN M. MARASIGAN**, University of the Philippines Los Baños,  
JUNE 2024. **ARGUSURE: A GEMINI-BASED CHROME EXTENSION FOR  
COUNTERARGUMENT GENERATION**

Major Professor: ASSOC. PROF. CONCEPCION L. KHAN

Filter bubble tends to cause bias and polarization. To alleviate its negative effects, it is important to utilize resources that offer various points of view. With this in mind, the study developed ArguSure. Powered by Google's multimodal LLM called Gemini, it is a counterargument generator Chrome extension designed to provide a way for users to conveniently explore counter viewpoints, especially the ones refuting what they see in digital spaces. It also has a web application for user, counterargument, and statistics viewing and management. To test and evaluate the usability of the application and the quality of the counterarguments it generates, the System Usability Scale (SUS) and LLM-based automatic evaluation were used, respectively. With a SUS score of 86.82 obtained from 11 respondents, ArguSure can be classified as an above-average in terms of user experience. Additionally, with mean scores of 4.55 to 4.98 obtained from automatic evaluation, the counterarguments are perceived to be clear, relevant, valid reasoning-wise, logically consistent, and effective overall. The findings also show that, compared to each criterion, the counterarguments are strongest in relevance and weakest in validity of reasoning.

**Index terms:** filter bubble, Chrome extension, counterargument generation, MERN, Gemini, prompting

# INTRODUCTION

## Background of the Study

Filter bubble, sometimes used interchangeably with the term Echo Chamber, is a situation in which we are surrounded by like-minded people and encounter only information that conforms to and reinforces our own beliefs (Bruns, 2019). Due to how the Internet and social media algorithms work, filter bubbles are prevalent in digital spaces (Pariser, 2018). The algorithms pick what we read and try to guess what we are interested in to make us more engaged. This tends to cause tribal thinking and bias as we rarely have to read something we do not agree with. One of the main problems with filter bubbles is they polarize people too much. Falling into one will cause us to have a biased perception as we will only be confined to a side of reality. To avoid this and reduce the effects of Filter bubbles, we should strive to maintain an impartial and objective stance when using the Internet and actively seek out and comprehend different types of information (S. Chen, 2023). Deliberately seek other viewpoints, especially from people we disagree with, and create a culture that encourages dissent. To help accomplish this and alleviate the effects of filter bubbles, the ArguSure Chrome extension is proposed.

ArguSure is a counterargument generator. It aims to provide a way for Internet users to conveniently explore counter viewpoints and be introduced to different, contradictory ideas. The idea of the application is simple. It will be a Chrome extension that, during the browsing experience, can provide counterclaims or counterarguments to a selected or inputted text. Chrome extensions are software programs that are installed in the Chrome browser that enhance the functionality of the browser (Chapagain, 2022). Currently, there are no Chrome extensions with the same functionality available. Similar applications, such as args.me (Wachsmuth et al., 2017), ArgumenText (Stab et al., 2018), and PerspectroScope (S. Chen, Khashabi, Callison-Burch, & Roth, 2019), are argumentative search engines.

They are search engines and most approaches done by these applications divide arguments into statements supporting or attacking the input topic or query. Unlike these applications, the ArguSure Chrome extension will focus on attacking the input and will be designed to be more accessible and easier to integrate into the user's browsing experience.

The foundation of ArguSure's counterargument generation will be the API provided by Google's multimodal LLM called Gemini (Team et al., 2024). Gemini is a family of generative AI models that lets developers generate content and solve problems (*Google AI for Developers*, n.d.). Its developer-focused API gives access to the latest generative AI models from Google and is designed to support both conversational and nonconversational use cases. To elicit proper counterarguments from this API, the process of creating well-structured prompts called Prompt design will be done as it is an essential part of ensuring accurate, high-quality responses from the generative AI model.

### **Statement of the Problem**

Filter bubble, also known as Echo Chamber, refers to the phenomenon where Internet customization effectively isolates individuals from diverse opinions or materials (Areeb et al., 2023). It is prevalent in digital spaces due to our biases and how the Internet and social media algorithm works. It creates an environment where users have less to zero exposure to different viewpoints and arguments that are against their beliefs. Filter bubbles have a profound societal impact as they affect individual cognition and polarize communities (M. Wang et al., 2024). To reduce the impact of these bubbles and broaden the scope of information, it is important for users to intentionally include diverse categories of topics in their preferences and actively seek out different types of information (S. Chen, 2023). With this, the study seeks to answer the following research questions:

1. What is the design and approach of the Chrome extension that can help Internet users conveniently explore counterviewpoints and be introduced to different, contradictory

ideas?

2. What technologies will be used in developing the Chrome extension and its corresponding web application?
3. What prompt designs will be used to elicit counterarguments from Gemini?

### **Significance of the Study**

To the extent of the researcher's knowledge, the result of this study will pave the way to the first counterargument generator in the form of Chrome extension. This Chrome extension can help in mitigating the effects of filter bubbles such as being exposed to only a select set of content over and over again causing people to have a biased and inaccurate perception of reality. This will be possible due to the purpose and nature of the Chrome extension - providing a way for Internet users to conveniently (as it is more accessible and easier to integrate into the user's browsing experience) seek and explore other viewpoints, particularly the ones countering what they are exposed to. This can help people enjoy the benefits of algorithms' personalized recommendations while also having easy access to diverse content. Additionally, aside from its intended uses, the Chrome extension paired with its corresponding web application can also be used in other ways such as by helping authors to better present their stance and aiding researchers to examine relevant citations (Orbach et al., 2020). After all, arguments are everywhere in academic writing.

The counterarguments that will be generated by the Chrome extension can also encourage people to invest more mental effort in processing the content they see while browsing the Internet. The reason for this is that LLMs tend to generate arguments that require higher cognitive effort than those produced by humans (Carrasco-Farre, 2024), and higher cognitive processing can promote engagement as readers may interpret the need for such cognitive investment as a sign of the argument's substance or importance (Kanuri, Chen, & Sridhar, 2018). On a different note, the study will also introduce prompt designs

to be used in LLMs, Gemini in particular, if we want them to generate counterarguments. Lastly, the results of the study can provide insights into the capabilities of Gemini API in producing proper arguments against an input text.

### **Objectives of the Study**

The general objective of the study is to develop a Chrome extension that can be used to conveniently explore arguments against what we see while browsing the Internet. Specifically, the objectives that the researcher set out to accomplish are:

1. To develop a counterargument generator Chrome extension that can be used through text selection and the context menu;
2. To develop a web application for users to view/organize generated counterarguments and for admins to view/organize application data and users;
3. To design prompts and utilize the services provided by Gemini API for input assessment and counterargument generation;
4. To evaluate the usability of the Chrome extension and its web application using the System Usability Scale (SUS); and
5. To evaluate the quality of the application's generated counterarguments using LLM-based automatic evaluation.

### **Scope and Limitations**

The focus of the study is the development of an application that can assess whether an input text is suitable for counterarguments and then automatically generate arguments against it if it is. This application does not necessarily aim to persuade or encourage users to change their views or beliefs. It just aims to let users explore different and contradicting ideas. It is also important to note that since the application uses Gemini, its

input assessment and counterargument generation capabilities are limited by the extent of the aforementioned LLM's capabilities. To give an example, the LLM is not safe from the occurrence of hallucinations. Hallucinations are when LLMs lie and fabricate non-existent facts or inappropriate information (Yao, Ning, Liu, Ning, & Yuan, 2023).

The application is in the form of a Chrome extension. It also has a corresponding web application for user, counterargument, and statistics viewing and management. It is only available online and does not support offline services. The Chrome extension will only be accessible through desktops with Google Chrome installed. Its corresponding web application, however, will be accessible using devices with web browsers such as computers, tablets, and smartphones.

### **Date and Place of the Study**

The study was conducted during the 2nd semester of the academic year 2023-2024 at the Institute of Computer Science, University of the Philippines Los Baños.

## **REVIEW OF RELATED LITERATURE**

### **Argument and Counterargument Generation using Generative AI technologies**

Generative Artificial Intelligence (AI) refers to computational techniques capable of generating seemingly new, meaningful content such as text, images, videos, or other data, often in response to prompts (Feuerriegel, Hartmann, Janiesch, & Zschech, 2023). Given its capabilities, it has diverse applications. Some of these, especially for computer science academic research, are explored in a paper by Garrido-Merchan (2023). In it, the ability of generative AIs such as ChatGPT to evaluate the quality of articles, summarize texts, and create critical analyses and counterarguments highlights how adaptable and effective it is at advancing scholarly study. As mentioned, one particular application of generative AI explored is counterargument generation. It can generate constructive counterarguments, which provide a simulated peer review experience. The paper argued that this helps researchers anticipate and respond to possible critiques, consequently enhancing the academic rigor of their work. By presenting opposing views and challenging accepted beliefs, generative AI promotes a more thorough and varied exploration of research topics.

When dealing with counterargument generation using generative AI, Large Language Models (LLMs) should be tackled. LLM is a specific type of generative AI that specializes in processing and generating text-based content. To audit the controlled counterargument capabilities of LLMs, Verma, Jaidka, and Churina (2024) evaluated three LLMs namely GPT-3.5, PaLM2, and Koala. From these LLMs, a new dataset of 32000 counterarguments to posts from the Reddit ChangeMyView dataset is generated. They called this new dataset Counterfire. Counterfire was used to evaluate LLMs' ability to create stylistic and evidence-based counterarguments. They found that on the one hand, the stylistic counterarguments still fall short of human persuasive standards, where people also preferred reciprocal to evidence-based rebuttals. They concluded that the

counterarguments produced by humans demonstrate a greater level of nuance and diversity in persuasive strategies. On the other hand, the LLMs show notable ability to integrate argument styles, especially in the "reciprocity" category, and to rephrase content with relevant evidence even with minimal lexical overlap. GPT-3.5 turbo, in particular, ranked highest in argument quality with strong paraphrasing and style adherence, especially in "reciprocity" style arguments. Also, based on the variations in rhetorical moves and user preferences, the counterarguments generated by LLMs comprise more innovative and convincing uses of evidence.

Providing insights on LLMs' argument and counterargument generation capabilities, a study conducted by Carrasco-Farre (2024) investigated the persuasion strategies of LLM-generated arguments and compared them with human-generated arguments. To accomplish this, a dataset of 1251 participants in an experiment was utilized. The measures used in the comparison were cognitive effort (lexical and grammatical complexity) and moral-emotional language (sentiment and morality). The study's findings indicated that LLMs generate arguments that require higher cognitive effort than those produced by humans, with more complex grammatical and lexical structures. When it comes to moral language, they find that LLMs demonstrate a substantial predisposition to engage more profoundly with it, utilizing both positive and negative moral grounds more frequently than humans. In terms of emotional content, on the other hand, no significant difference was found between arguments produced by LLMs and humans, which is different from previous research. As mentioned, LLM arguments require higher cognitive effort due to increased grammatical and lexical complexity. One might think that this would cause a lower persuasion level. The result, however, suggests otherwise. The complexity might encourage deeper cognitive engagement (Kanuri et al., 2018), prompting readers to invest more mental effort in processing the arguments, which is one of the main objectives of the paper. This study by Carrasco-Farre (2024), together with the preceding reviewed studies by Garrido-Merchan (2023) and Verma et al. (2024), support the paper's use of Gemini in



generating counterarguments as they show that generative AIs, particularly LLMs, have the capability to generate reliable arguments and counterarguments.

## **Prompting**

To effectively converse with LLMs, prompt engineering is an increasingly important skill set as it plays a pivotal role in unleashing their capabilities. Prompt engineering involves strategically designing task-specific instructions, referred to as prompts, to guide model output without altering parameters (Sahoo et al., 2024). Several papers, such as the studies done by Liu et al. (2021), White et al. (2023), and B. Chen, Zhang, Langrené, and Zhu (2023), explored and evaluated different prompt designs and techniques in increasing the LLM efficacy without modifying the core model parameters. Such techniques include one-shot, few-shot, and chain-of-thought prompting. An example of a study that utilized some of these techniques is the paper by Majer and Šnajder (2024). They studied the predictive and calibration accuracy of zero- and few-shot LLM prompting for claim detection (CD) and checkworthiness (CW). They experimented with five CD/CW datasets from diverse domains, each utilizing a different worthiness criterion, and investigated two key aspects: (1) how best to distill factuality and worthiness criteria into a prompt and (2) what amount of context to provide for each claim. The findings show that optimal prompt verbosity depends on the domain, adding context does not enhance performance, and confidence scores can be directly used to produce reliable check-worthiness rankings. These findings show the potential of using LLMs for claim check-worthiness detection with minimal prompt engineering.

The preceding paper used LLM prompting techniques to identify factual and check-worthy claims. Hu, Chan, and Yin (2023), on the other hand, used them to generate arguments. They proposed a novel framework, called AMERICANO, with agent interaction for argument generation. This framework first produces an argument draft through a generation agent, and then iteratively produces feedback and revises the draft

through an evaluation agent and refinement agent, respectively. This approach was inspired by recent chain-of-thought prompting that breaks down a complex task into intermediate steps. All parts of the model are implemented leveraging LLMs with zero-shot prompting, with a subset of propositions collected from Reddit/CMV dataset. To validate the model output, they leveraged both LLM-based automatic evaluation and human evaluation. Both evaluations show that AMERICANO can generate more coherent and persuasive results than end-to-end and CoT prompting, and it can also generate more diverse and rich results than baseline methods. Given the promising results of these studies that utilized LLM-prompting to achieve their objectives, the studies by Majer and Snajder (2024) and Hu et al. (2023) can be used as the basis for prompts that will be fed to Gemini for it to generate high-quality and coherent counterarguments.

### **Related Software Applications**

Currently, there are no Chrome extensions with the same functionality available. There are, however, similar applications in the form of argumentative search engines. Argumentative search engines, as opposed to regular online search engines, must find the most pertinent arguments given the query terms and document collections to search in (Daxenberger, Schiller, Stahlhut, Kaiser, & Gurevych, 2020). These search engines usually accept a topic as input, then provide arguments for and against it. args is one of these search engines. This software was a result of the study conducted by Wachsmuth et al. (2017), where they develop an argument search framework for studying search-related questions in the context of computational argumentation. Given any free text query, args looks for and ranks arguments related to it in an initial and freely accessible index of nearly 300k arguments crawled from reliable web resources. This search engine, along with the framework used by the study, is intended as an application-oriented environment for collaborative research on computational argumentation.

Another example of an available application related to Counterargument Generator in

terms of functionality is ArgumenText. Similar to args, it is also an argumentation-based search engine presented by Stab et al. (2018). It is capable of retrieving “pro” and “con” arguments for a given controversial topic from heterogeneous sources. The study analyzed ArgumenText by comparing the high-ranked arguments it generates to arguments from debate portals. The results show that it covers 89% of arguments found in expert-curated lists of arguments and that it is even capable of finding additional valid arguments. Despite the high coverage, the precision of the system can still be improved by implementing more sophisticated deep learning architectures and other argument ranking methods.

PerspectroScope, similar to the previously discussed related software applications, is also an argumentative search engine. PerspectroScope was presented by S. Chen et al. (2019) with the objective of letting users investigate various angles that might address facets of the current problem. It is a web-based interface that lets users query a discussion-worthy natural language claim and then explore and visualize the perspectives that support or counter the claim, along with evidence for each perspective. Users can also send feedback to the system if they think some corrections are needed. Behind PerspectroScope’s interface lies retrieval engines and learned textual entailment-like classifiers. This is to balance the speed and quality of the perspective generation. PerspectroScope is a powerful tool, however, there were still many issues and limitations that are not addressed such as its potential of having non-exhaustive perspectives or unreliable identified sources. Also, similar to the previously discussed software applications, it is not integrated into the browsing experience as users need to go to the website first to explore different arguments. This feature does not support one of the main objectives of the paper, which is to give users a more convenient way of exploring the opposite side of what they see while browsing the Internet.

## METHODOLOGY

### Development Tools

The Chrome extension (with its corresponding web application) was developed on a machine with the following specifications:

1. Operating system: Windows 11 Home Single Language 64-bit
2. Processor: 11th Gen Intel(R) Core(TM) i5-1155G7 @ 2.50GHz 2.50 GHz
3. Memory: 24.0 GB (23.8 GB usable)

The following software development tools and technologies were used for the development of the Chrome extension and its web application.

#### 1. Environment

##### (a) *Visual Studio Code 1.88.1*

A feature-rich source code editor that served as the main environment for developing the application.

##### (b) *Google Chrome*

A web browser developed by Google which was used as the platform for developing the application.

#### 2. Technologies

##### (a) *Node.js*

An asynchronous event-driven JavaScript runtime that is designed to build scalable network applications. This was used as the runtime environment of the application.

##### (b) *Express.js*

A minimal and flexible minimalist web framework for Node.js that provides a robust set of features for web and mobile applications. This handled the back-end component of the application.

(c) *MongoDB*

A cross-platform, document-oriented, open-source NoSQL database management program that utilizes JSON-like documents with optional schemas. This served as the main database of the application.

(d) *React*

A free and open-source front-end JavaScript library that lets developers build user interfaces out of individual pieces called components. This was used to develop the client side of the application.

(e) *Tailwind CSS*

A utility-first CSS framework packed with classes that can be composed to build any design, directly in the markup. This was used to style the UI of the application.

(f) *Vite*

An opinionated modern front-end build tool for modern web projects. This was used to set up the development environment for the Chrome extension and its web application.

(g) *Gemini API*

An Application Programming Interface (API) that gives developers access to Gemini, a series of multimodal generative AI models developed by Google. This was used in the application's counterargument generation.

## Features

This section explains the features associated with the the application's Chrome extension and its corresponding web application.

### Chrome Extension

#### 1. *Input Mechanism*

Users can input a text to be argued against through these two methods: (1) by highlighting a text on the screen and then using the context menu and (2) by typing the text in the text field found in the popup.

#### 2. *Counterargument Generation*

The Chrome extension can generate and display counterarguments (with summary, body, and sources) for the input claim/argument. The way counterarguments will be displayed depends on which input mechanism the user chooses.

#### 3. *Like/Dislike Counterargument*

Users can like or dislike the counterarguments generated by the Chrome extension.

#### 4. *Save/Unsave Counterargument*

Users can save or unsave the counterarguments generated by the Chrome extension. Instead of just saving (default), users can also opt to save a counterargument to a particular topic/s.

#### 5. *Create Topics*

Users can create topics that can serve as the categories for the saved counterarguments.

#### 6. *Web Application Redirection*

Users can be redirected to some of the web application's sections such as the home page and profile page.

## **Chrome Extension's Web Application**

### *1. Sign Up*

Users are required to create an account to use the application. Necessary information such as username and email is needed. They can also opt to register using their Google account.

### *2. Sign In*

Users are required to sign in themselves before using the application. They can also opt to sign in using their Google account.

### *3. Counterargument Generation*

The web application can generate and display counterarguments (with summary, body, and sources) for the input claim/argument. Users can use this on the home page.

### *4. Like/Dislike Counterargument*

Users can like or dislike the counterarguments generated by the web application.

### *5. Save/Unsave Counterargument*

Users can save or unsave the counterarguments generated by the web application. Instead of just saving (default), users can also opt to save a counterargument to a particular topic/s.

### *6. Topics*

Users can view, create, edit, and delete topics that can serve as the categories for the saved counterarguments.

### *7. View Counterarguments*

Users can view counterargument generation history. They can also view all liked, disliked, and saved counterarguments in the web application.

#### 8. *Search*

Users can search counterarguments. Every section that displayed counterarguments such as the likes and saved section has a search feature.

#### 9. *Profile*

Users can view and update their profile information such as username and profile picture.

#### 10. *Admin Page Dashboard*

Admins can view the application statistics such as the total number of users, liked and disliked counterarguments, and past month's data. Only admins have access to this feature.

#### 11. *Admin Page Users Section*

Admins can view all the application users and some of their information. They can also delete all the users. Only admins have access to this feature.

### **Use Case**

The diagram in Figure 1 is the use case for the application. It shows what users and admins should do to access the features discussed in the previous section. The input mechanism depends on where the user uses the application: (1) highlight/edit in the Chrome extension's context menu, and (2) type in the text field in the Chrome extension's popup and web application's homepage. Admins have access to every feature users have access to. Only admins, however, can view the application statistics and view or delete registered users.



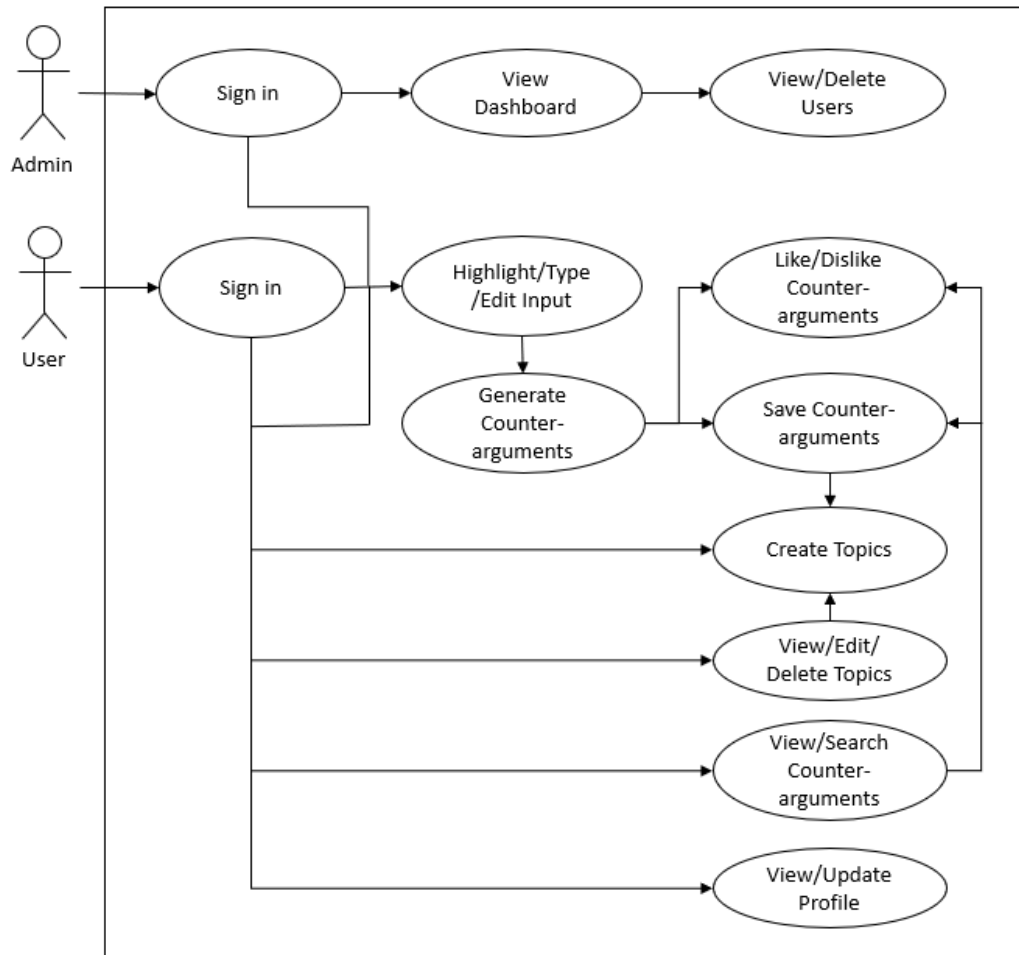


Figure 1. Use Case Diagram for the Counterargument Generator Application

### Counterargument Generation Flowchart

The flowchart in Figure 2 shows how a random text input is processed for counterargument generation. Before the counterargument generation, input assessment will be done first to determine if the input is suitable for counterarguments. If it is, the counterargument generation will start. Then, the generated counterarguments will be checked if they passed the safety filters. If they do, it will only be then that the counterarguments will be displayed to the user.

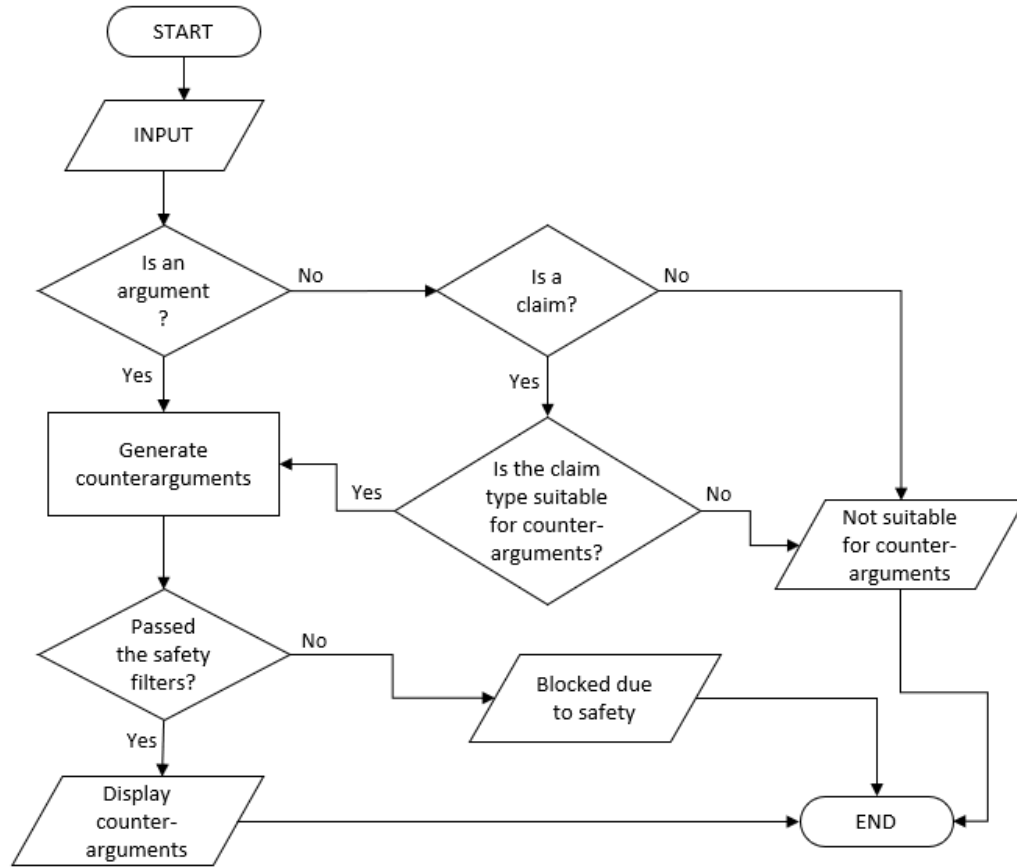


Figure 2. Counterargument Generation Flowchart

## Database

The database consists of two collections: (1) users and (2) counterarguments. All the user information is stored in the users collection while all the counterarguments generated are stored in the counterarguments collection. The fields in users are username, email (unique), password, profile picture, saved counterarguments, and admin classification. The fields in counterarguments are user ID, input claim, summary, body, source, and liked status. Each collection has a unique ID. In the users' saved counterarguments field, only the ID of counterarguments is contained. The user who generated each counterargument is determined by the user ID in counterarguments collection.

## **Application Testing and Evaluation**

To test and evaluate the usability of the Chrome extension and its web application, the System Usability Scale (SUS) was used. SUS is a widely used standardized questionnaire for the assessment of perceived usability (Lewis, 2018). In its standard form, the questionnaire has 10 five-point items with alternating positive and negative tone. This questionnaire was answered by human evaluators after they tested different features and functionalities of the application.

On the other hand, to test and evaluate the quality of the counterarguments, LLM-based automatic evaluation was used. In evaluating the quality of texts, LLMs demonstrate a potential to be an alternative to human experts as LLM evaluation results are consistent with human expert evaluation results (Chiang & yi Lee, 2023). This means that the texts rated higher by human experts were also rated higher by the LLMs.

Specifically, the LLM family used in the study to evaluate counterarguments was GPT-3.5, which was accessed through ChatGPT. Despite several limitations, ChatGPT (GPT-3.5) exhibited potential reliability and credibility for argumentation feedback. Especially for shorter arguments, such as the counterarguments generated by the application, ChatGPT possesses a fundamental capability to provide feedback on arguments (L. Wang et al., 2024).

## **RESULTS AND DISCUSSION**

### **Development**

#### **Development Environment**

In developing both the Chrome extension and its corresponding web application, Vite was used to configure the development environment for React. It significantly helped the development process as it provided almost instant and precise updates without reloading the page or blowing away the application state. To initialize it, Node Package Manager (npm) was used. npm was also used to install and manage modules and packages needed as dependencies of the application.

For the Chrome extension's source code, a boilerplate (Silkstone, 2023) was used as the starting point. This boilerplate is designed for modern web extensions that use React and Tailwind. It also uses Vite to build. Most of the codes for the Chrome extension are in the service workers and popup module. The service workers handle the context menu and window popup (appears when the context menu is used) functionality, while the popup module contains the code for the Chrome extension's popup (appears when the extension icon is clicked).

#### **Gemini API**

The foundation of the application's counterargument generation is Gemini API. Gemini is Google's multimodal LLM, which is the first to outperform human experts on MMLU (Massive Multitask Language Understanding), a popular method for testing the knowledge and problem-solving abilities of AI models (Team et al., 2024). Gemini models can enable new approaches in areas like education, everyday problem-solving, information summarization, extraction, and creativity. In the study's case, it is particularly

used to generate arguments against a claim or another argument. Suited to this purpose, the model variation used is Gemini Pro as it inputs/outputs text and can handle zero, one, and few-shot tasks. This model is performance-optimized in terms of cost as well as latency which delivers significant performance across a wide range of tasks. The rate limit for this model is 60 requests per limit (RPM).

In Gemini API, there are options to control content generation. This is done through configuring the model parameters and safety settings. The model parameter values used in the application are the default values provided by the API - max output tokens of 2048 (100 tokens correspond to roughly 60-80 words), temperature of 0.9, topK of 1, and topP of 1. Temperature controls the degree of randomness in the token selection while topK and topP change how the model selects tokens for output. For the safety settings, the developer adjusted all categories (harassment, hate speech, sexually explicit, and dangerous) so that they only block output when there is a high probability of unsafe content.

## **Prompting**

For Gemini to produce counterarguments, the right input text, also called prompt, must be crafted accordingly. The problem is, given that the behavior of generative models is largely opaque even to the model trainers, we often cannot predict in advance what types of prompt structures will work best. Still, there are ways to increase the efficacy of generative models. In the study, for instance, to increase the chance that Gemini will generate proper counterarguments, some of the concepts of prompt design (also called prompt engineering or simply prompting) are utilized. Prompt design is the art and science of figuring out the right wording to get generative models to do what we want. Some prompt design strategies used in counterargument generation are defining the task to perform, specifying constraints, defining the format of the response, breaking down prompts into simple components, and zero-shot prompting. The prompts used for input assessment and counterargument generation are shown in Figures 3 to 6.

Categorize the sentence "<input>" into seven categories:

1. Personal experience (PE): Claims that aren't capable of being checked using publicly-available information, e.g. "I can't save for a deposit."
2. Quantity in the past or present (Q): Current value of something e.g. "1 in 4 wait longer than 6 weeks to be seen by a doctor." Changing quantity, e.g. "The Coalition Government has created 1,000 jobs for every day it's been in office." Comparison, e.g. "Free schools are outperforming state schools.". Ranking, e.g. "The UK's the largest importer from the Eurozone."
3. Correlation or causation (CC): Correlation e.g. "GCSEs are a better predictor than AS if a student will get a good degree." Causation, e.g. "Tetanus vaccine causes infertility." Absence of a link, e.g. "Grammar schools don't aid social mobility."
4. Current laws or rules of operation (CLO): Declarative sentences, which generally have the word "must" or legal terms, e.g. "The UK allows a single adult to care for fewer children than other European countries." Procedures of public institutions, e.g. "Local decisions about commissioning services are now taken by organisations that are led by clinicians." Rules and changes, e.g. "EU residents cannot claim Jobseeker's Allowance if they have been in the country for 6 months and have not been able to find work."
5. Prediction (P): Hypothetical claims about the future e.g. "Indeed, the IFS says that school funding will have fallen by 5% in real terms by 2019 as a result of government policies."
6. Other type of claim (OTC): Voting records e.g. "You voted to leave, didn't you?" Public Opinion e.g. "Public satisfaction with the NHS in Wales is lower than it is in England." Support e.g. "The party promised free childcare" Definitions, e.g. "Illegal killing of people is what's known as murder." Any other sentence that you think is a claim.
7. Not a claim (NAC): These are sentences that don't fall into any categories and aren't claims. e.g. "What do you think?.", "Questions to the Prime Minister!"

Strictly use only one of the 7 labels (PE, Q, CC, CLO, P, OTC, NAC), do not provide any additional explanation.

Figure 3. Prompt to determine the claim type of the input

Strictly yes or no, is "<input>" an argument? Please note that an argument is a coherent series of reasons, statements, or facts intended to support or establish a point of view.

Figure 4. Prompt to determine if the input is an argument

Strictly yes or no, is "<input>" a claim? Please note that a claim is an assertion open to challenge.

Figure 5. Prompt to determine if the input is a claim

Please provide one argument against "<input>" strictly with summary (in paragraph form labeled as **\*\*Summary:\*\***), body (in paragraph form labeled as **\*\*Body:\*\***), and source (in bullet points labeled as **\*\*Source:\*\***) as the format. The argument should be well-structured and organized in a coherent manner.

Please make sure that the argument will refute "<input>" and not support it.

Please provide another argument against "<input>" strictly with summary (in paragraph form labeled as **\*\*Summary:\*\***), body (in paragraph form labeled as **\*\*Body:\*\***), and source (in bullet points labeled as **\*\*Source:\*\***) as the format. The argument should be well-structured and organized in a coherent manner.

Please make sure that the argument will refute "<input>" and not support it.

Again, please provide another argument against "<input>" strictly with summary (in paragraph form labeled as **\*\*Summary:\*\***), body (in paragraph form labeled as **\*\*Body:\*\***), and source (in bullet points labeled as **\*\*Source:\*\***) as the format. The argument should be well-structured and organized in a coherent manner.

Please make sure that the argument will refute "<input>" and not support it.

Figure 6. Prompts for Counterargument Generation

## Chrome Extension and Web Application

1. Sign up - To use the application, users need to create an account first. They need to supply the following input fields: username, email, password, and confirm password. Email should be unique across all users. They can also opt to sign up using their Google account. If the sign-up process is successful, they will be navigated to the sign-in page. However, if they opt to use their Google account and it is already registered, they will be signed in and proceed to the homepage.

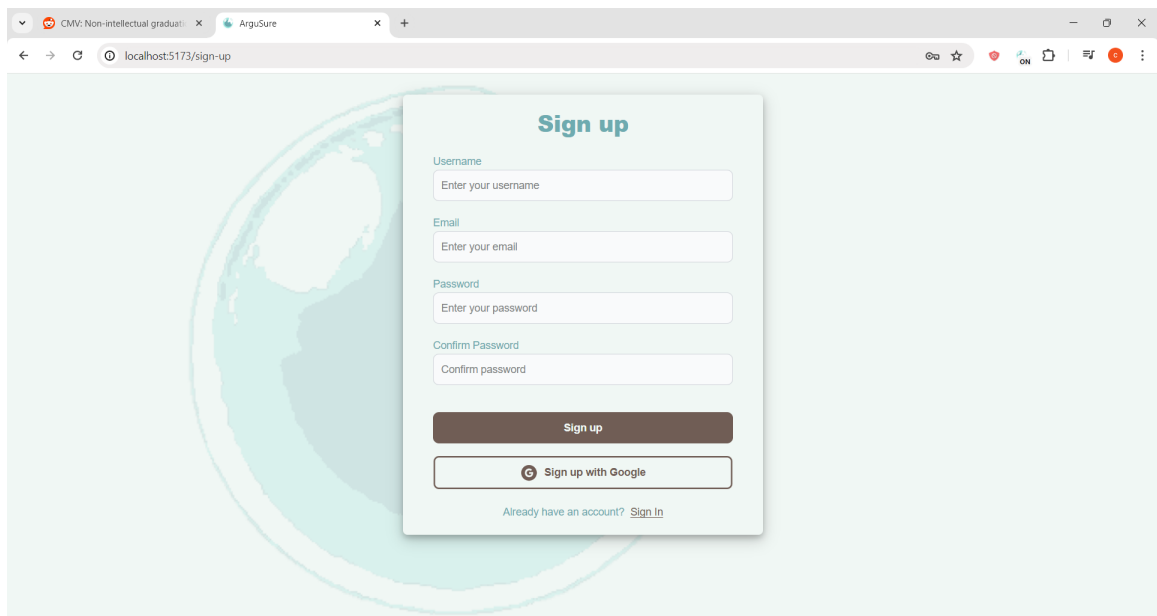


Figure 7. Sign up Page

2. Sign in - Users who already have an account need to sign in first before using the application. They can do this by submitting their email and password. They can also opt to sign in using their Google account. Successful authentication will redirect them to the homepage.
3. Landing Page - The landing page, which has the app name, logo, and description, can be found in different sections depending on where the users open the application. If they use the context menu, the window popup will display the landing page if they



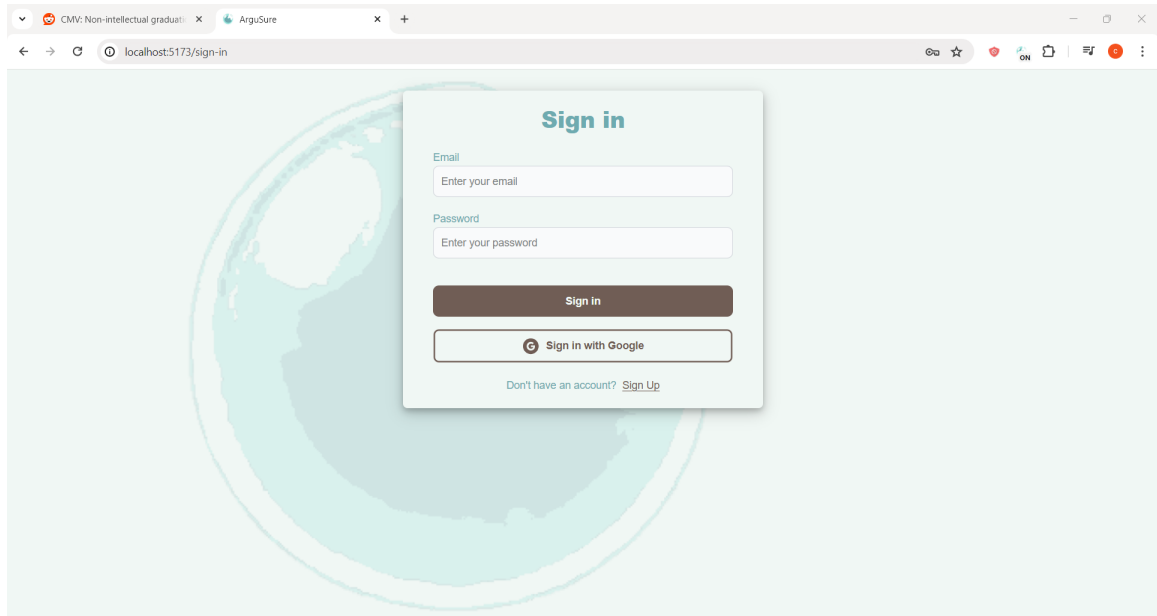


Figure 8. Sign in Page

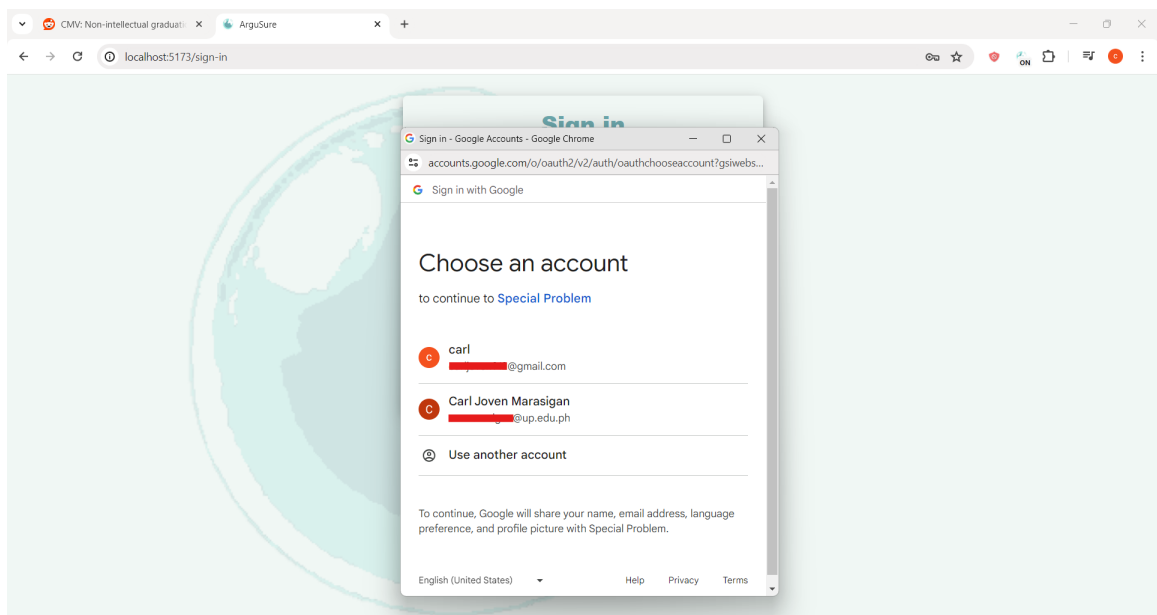


Figure 9. Sign in or Sign up using Google Account

are not signed in. If they click the Chrome extension icon, the landing page will be displayed in the Chrome extension pop. Lastly, if they go to the web application, the landing page can be found on the homepage.

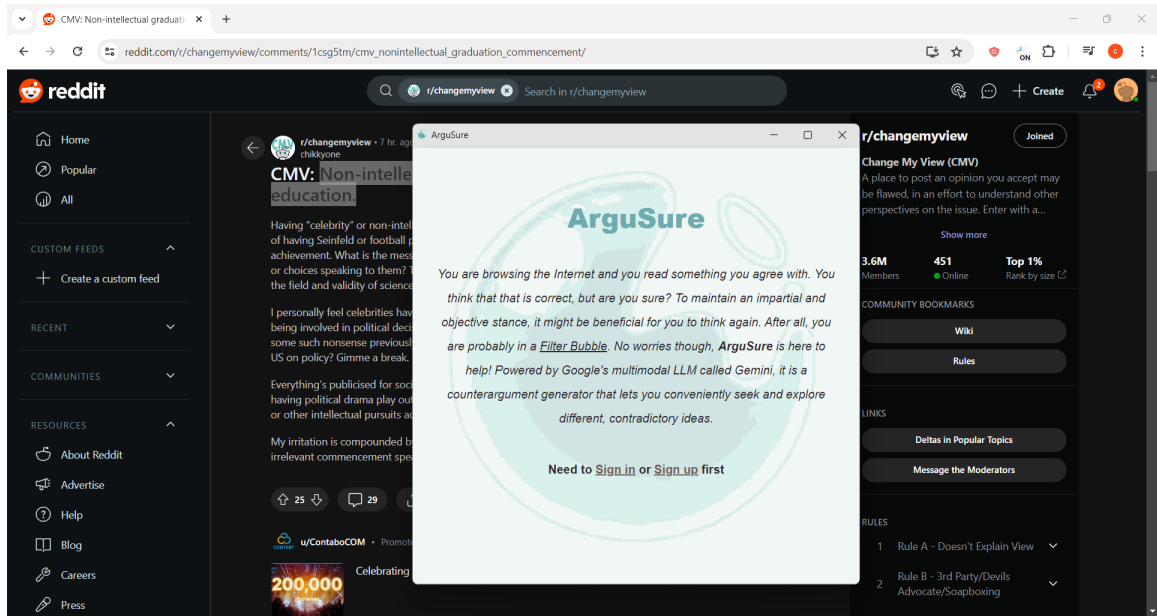


Figure 10. Window Popup Landing Page

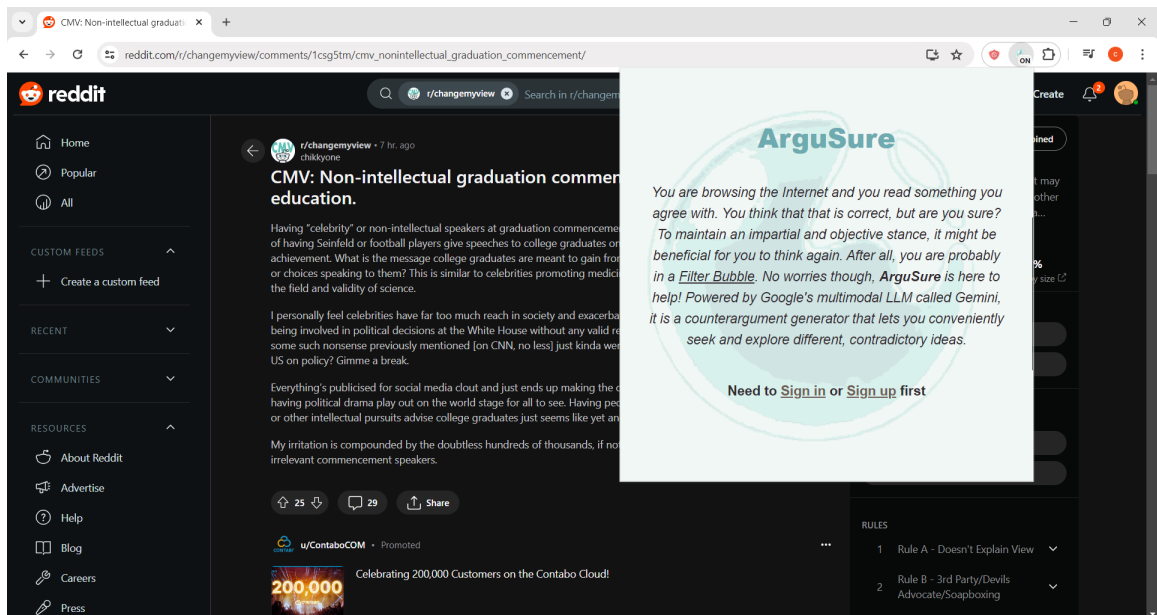


Figure 11. Chrome Extension's Popup Landing Page

4. Input Mechanisms - Users can input a text to be assessed and argued against through these three methods: (1) by selecting/highlighting a text on the screen and then using the context menu, (2) by typing the text in the text field found in the Chrome extension popup, and (3) by typing the text in the text field found in the

homepage of the web application. Note that in the first method, users can still edit the selected/highlighted text. The text input has a character limit of 500.

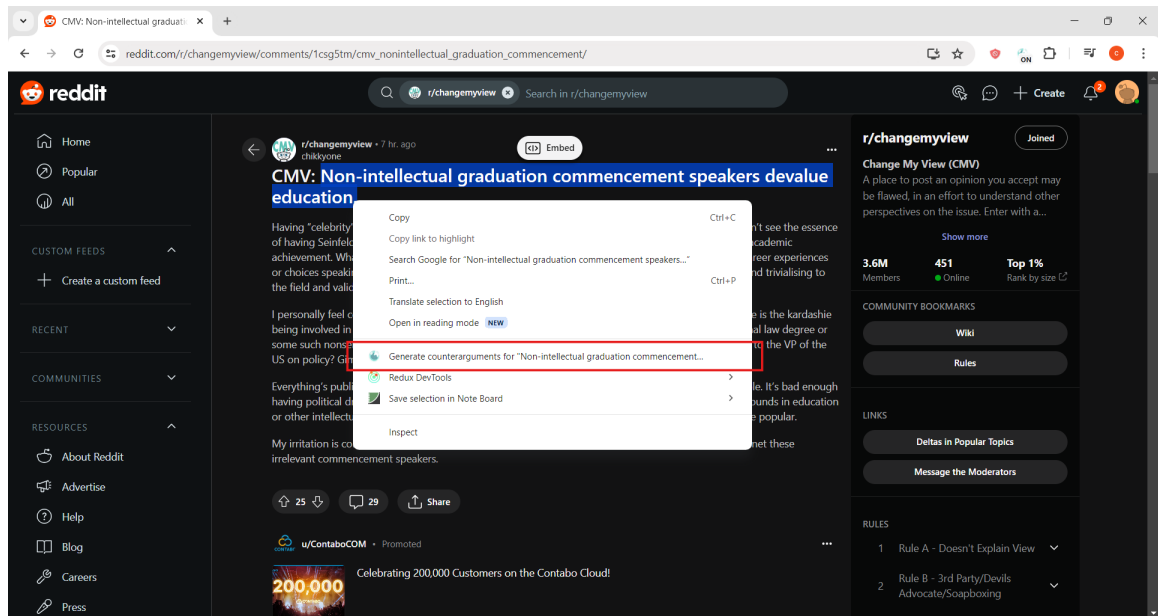


Figure 12. Context Menu

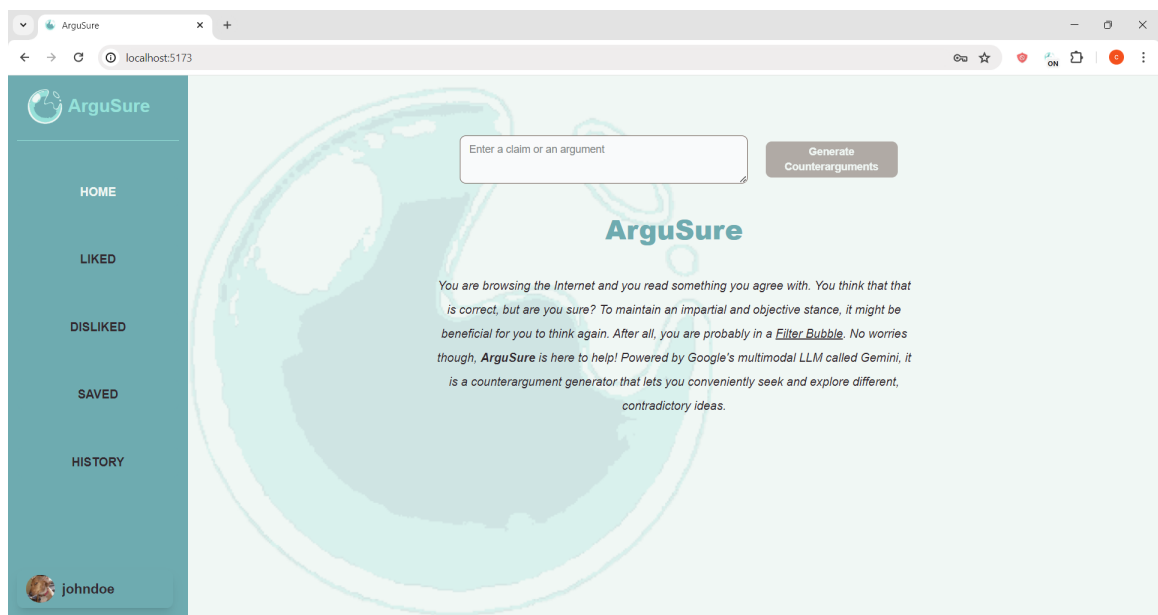


Figure 13. Home page

5. Counterargument Generation - The application is a counterargument generator.

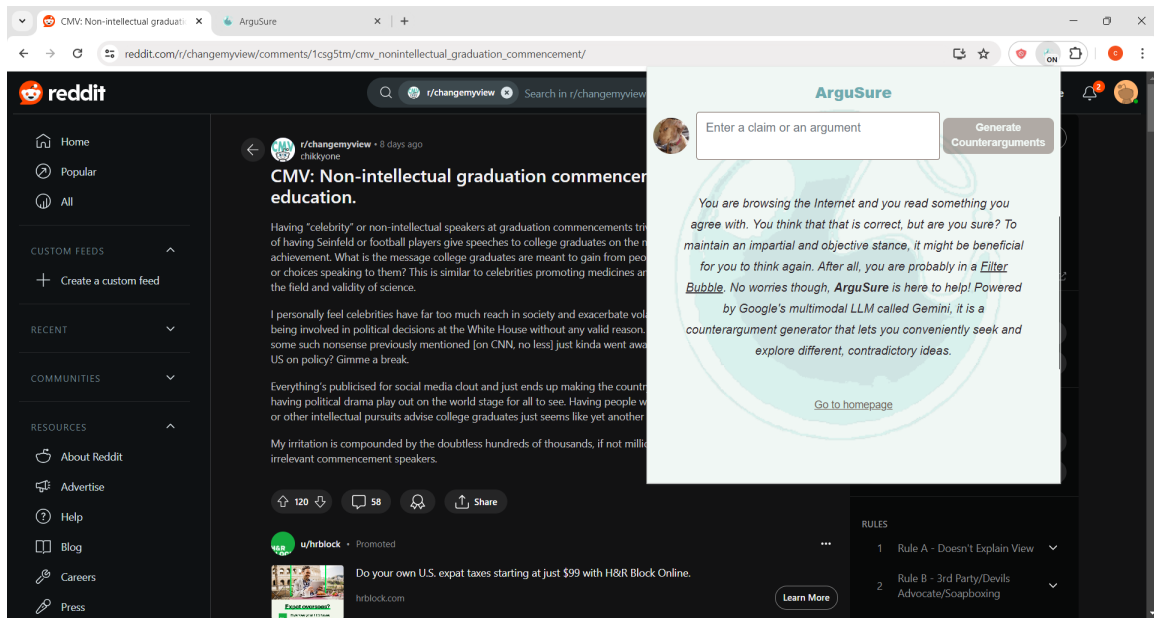


Figure 14. Chrome Extension's Popup

Before it generates counterarguments though, it will assess if the input is suitable for counterarguments first. If it is, the counterargument generation will start and three (3) counterarguments will be generated and displayed. Each counterargument contains a summary, body, and source/s. All of these are not displayed at first as they may be too long. However, users can opt to read all parts of the counterargument by clicking the "Read more" button. The users also have the option to "Regenerate" if they want to repeat the counterargument generation process.

6. Like/Dislike Counterarguments - Users can like or dislike generated counterarguments as they see fit. Aside from these actions being recorded for users' viewing, these will also be reflected on the admin dashboard for admins to have additional insights about the application.
7. Save Counterarguments - Users can save generated counterarguments. If a user saves a counterargument through the "Saved" button, it will be saved in the default section where all saved counterarguments, regardless of their topic, are displayed. On the other hand, if a user saves a counterargument through the "Save to" button, it will be

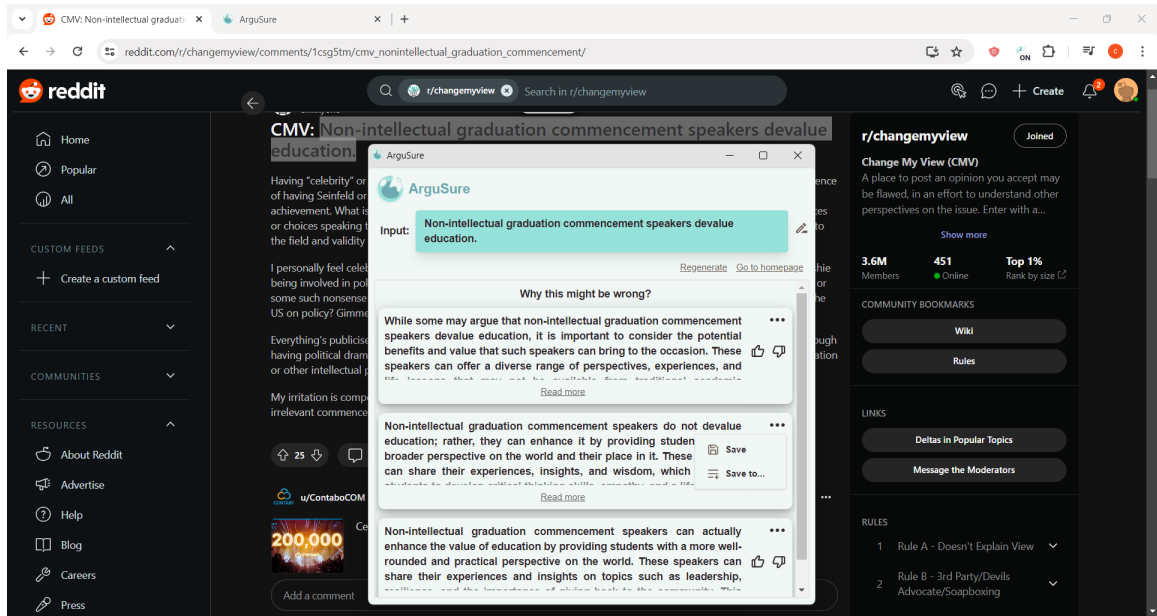


Figure 15. Window Popup with Generated Counterarguments

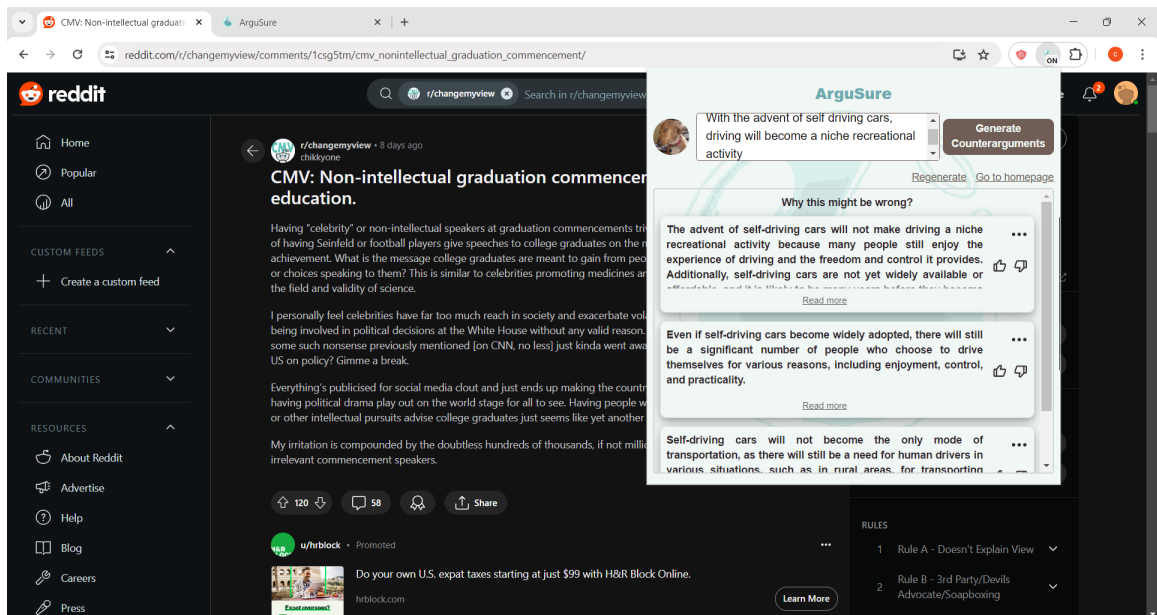


Figure 16. Chrome Extension's Popup with Generated Counterarguments

saved under the default section and topics the user selected. Users can also unsave counterarguments and remove them from a particular topic.

8. Topics - Users can view, create, rename, and delete topics. Essentially, topics



Figure 17. Home page with Generated Counterarguments

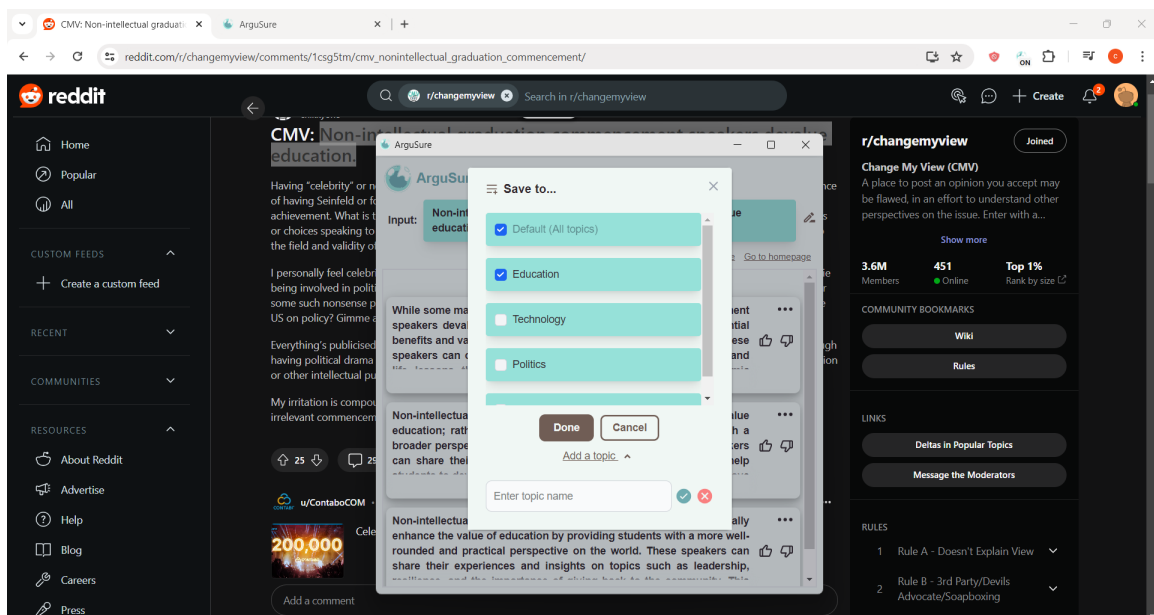


Figure 18. Save to Modal

are the categories or groupings of the saved counterarguments. Users can save counterarguments under them as they see fit. This feature helps users to organize the counterarguments they generated. Users can view or create topics both in the "Save to" modal and in the "Topics" section. Renaming or deleting topics, however,

is only accessible in the "Topics" section. Deleting a topic will also remove all counterarguments under it.

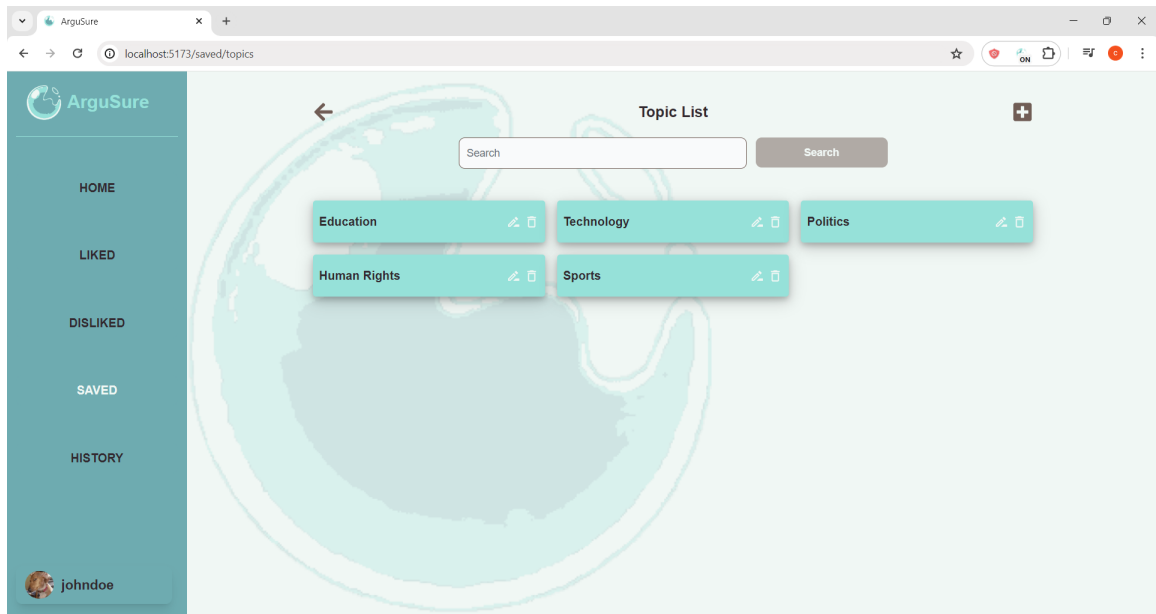


Figure 19. Topic List Page

9. Counterargument Viewing - Users can view all the counterarguments they generated both through the Chrome extension and the web application on the History page. They can view liked counterarguments on the Liked page, disliked counterarguments on the Disliked page, and saved counterarguments on the Saved page. Through the Saved page, users can navigate to the Topic List page if they want to view counterarguments saved under a particular topic. In all of these pages, counterarguments are sorted from most to least recent. Additionally, at most nine (9) counterarguments are only displayed at first. Users can just click "Show more" if they want to see more counterarguments.
10. Search - Users can search on every page where generated counterarguments are recorded and displayed. This search feature uses the input and all parts of the counterargument. This means that a counterargument will be included in the search results if the search term can be found in its input text, summary, body, or source/s.

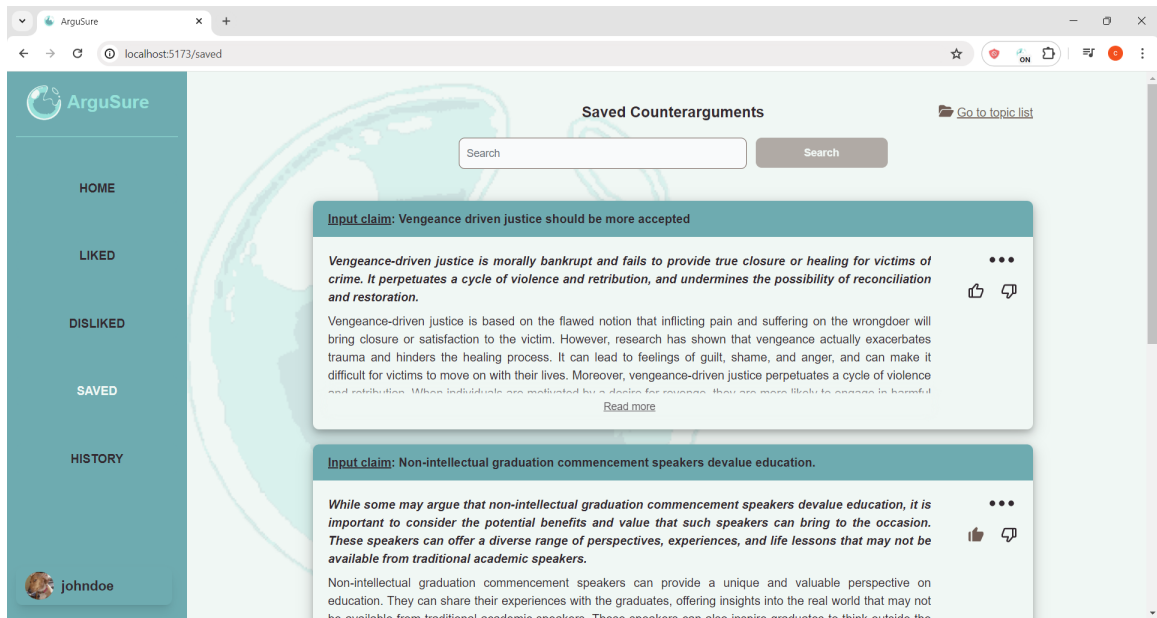


Figure 20. Saved Counterarguments Page

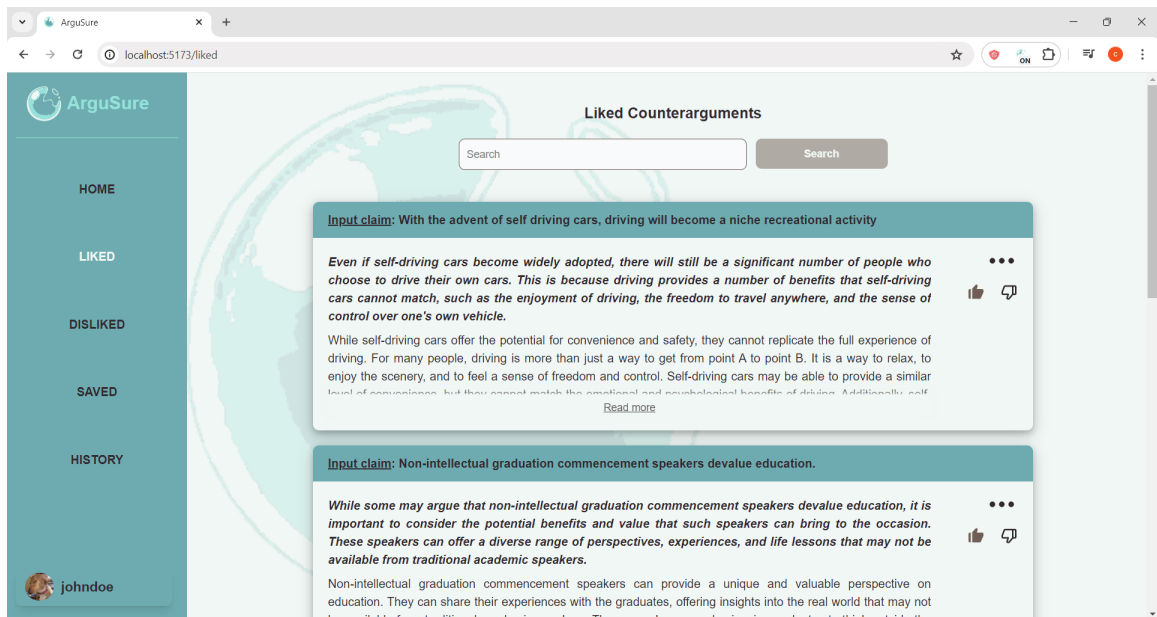


Figure 21. Sample Page for Counterarguments Viewing (Liked Counterarguments Page)

11. Profile - Users can view and update their profile information. They can view their username, email, and profile picture on the Profile page. On the same page, they can also update their username, profile picture, and password. Emails cannot be updated. Updating any profile information requires users to submit their current password.



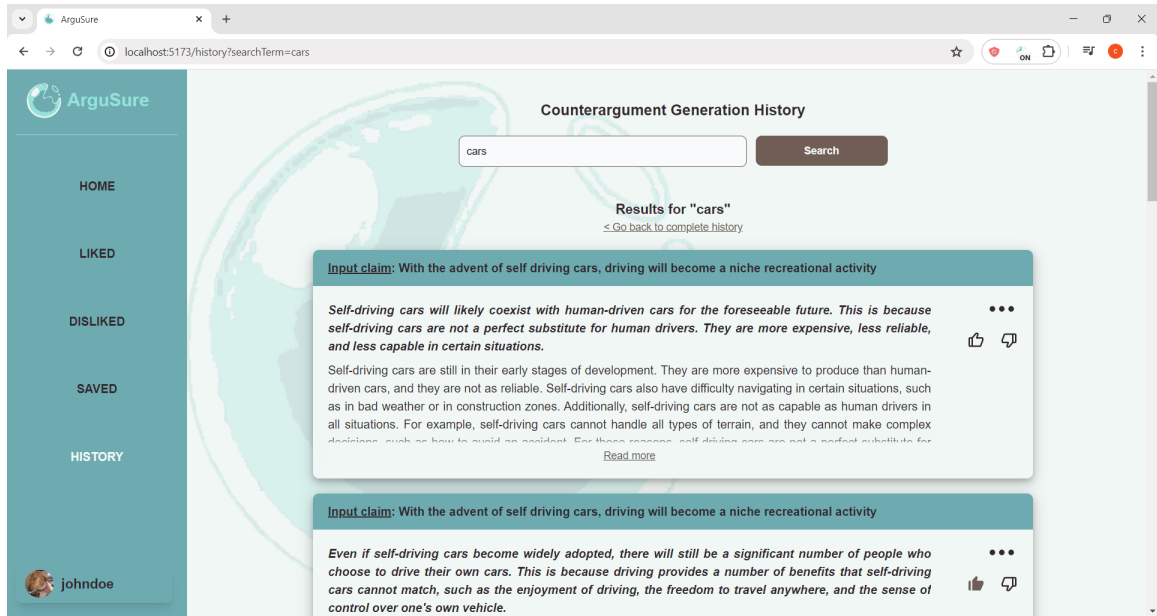


Figure 22. Search Results Sample

The Profile page also has a "Delete Account" button if users want to delete their account.

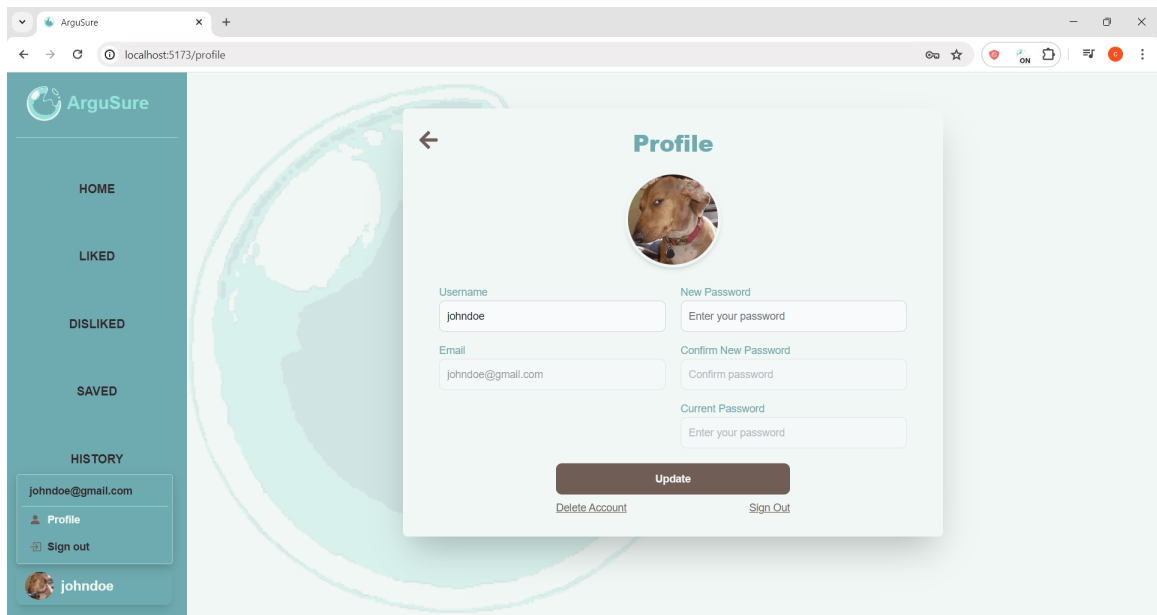


Figure 23. Profile Page

12. Admin Page Dashboard - Admins have access to every feature users have access to.

Only admins, however, can access the admin page dashboard. The dashboard shows the total number of users, generated counterarguments, and saved counterarguments. It also shows the total number of liked and disliked counterarguments, their ratio in percentage and pie chart form, and the application’s past month statistics.

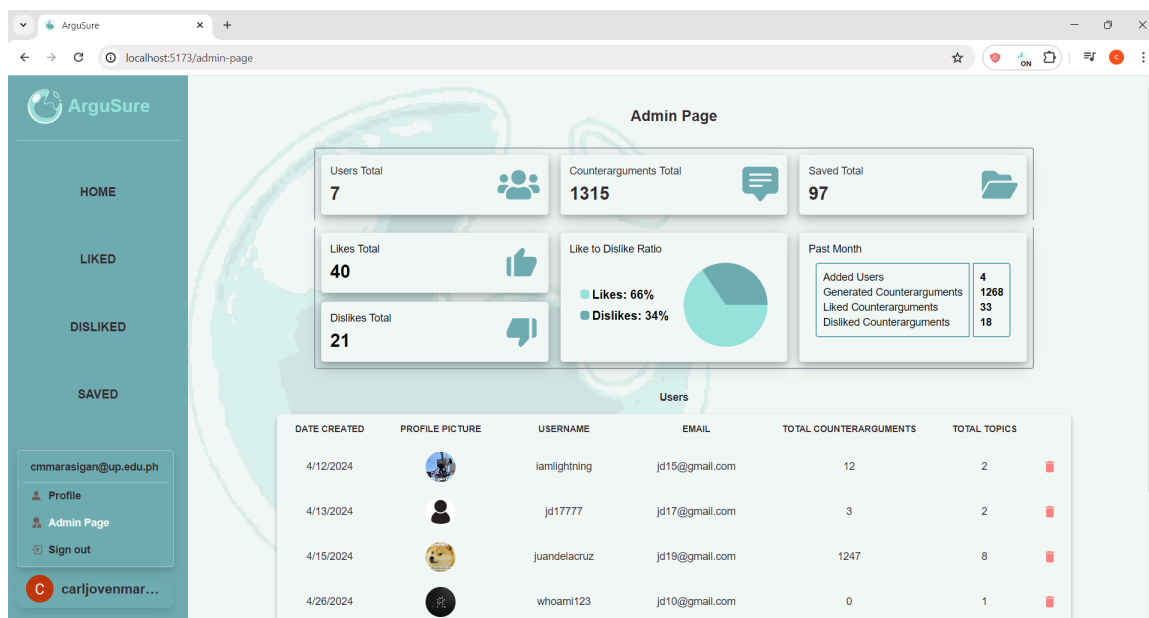


Figure 24. Admin Page Dashboard

- Admin Page Users Section - On the admin page, admins can also access the table of all users. This table includes each user’s date creation, profile picture, username, email, counterarguments generated total, and topics created total. Admins also have the privilege to delete users.

## System Usability Testing

The usability of the Chrome extension and its web application was tested by 11 UPLB students using the System Usability Scale (SUS). They were tasked to perform different features and functionalities of the application. For the feature that requires the user to select an input from the Internet, respondents were tasked to browse one of Reddit’s communities

DATE CREATED	PROFILE PICTURE	USERNAME	EMAIL	TOTAL COUNTERARGUMENTS	TOTAL TOPICS
5/16/2024		iamlightning	jd23@gmail.com	0	0
5/16/2024		user123	user123@gmail.com	0	0
5/16/2024		jd22222	jd22@gmail.com	0	0
5/16/2024		jd212121	jd21@gmail.com	0	0
5/16/2024		johndoe	johndoe@gmail.com	21	5
4/27/2024		carl4303	carl4303@gmail.com	0	0
4/27/2024		carljoenmarasigan8571	carljoenmarasigan@up.edu.ph	0	0
4/26/2024		whoami123	jd10@gmail.com	0	1
4/15/2024		juandelacruz	jd19@gmail.com	1247	8
4/13/2024		jd17777	jd17@gmail.com	3	2

Figure 25. Admin Page Users Section

called *changemyview*. This Reddit community suited the purpose of the task as it is filled with posts containing claims and arguments. Once the respondents were done with all the tasks given by the researcher, a SUS survey was administered to them through Google Forms. The survey utilizes a 10-item questionnaire with a five-point scale ranging from "Strongly Disagree" to "Strongly Agree." The SUS statements from the questionnaire are:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very awkward to use.

9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

After the collection of responses, the individual scores and mean score were computed. As shown in Table 1, the application obtained a mean score of 86.82 out of 100. Based on the SUS standards, this score corresponds to an "A+" grade rating, indicating an above-average in terms of user experience. Additionally, suggestions or comments were also asked to the respondents. The suggestions and comments made were about how user-friendly the application is, how the application can be of help, and recommended actions or improvements to the application.

Respondent	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
1	4	1	5	1	4	2	2	2	4	1	80
2	4	4	5	2	5	3	5	1	4	2	77.5
3	5	1	4	1	5	1	5	1	5	1	97.5
4	5	4	5	3	4	1	5	1	4	1	82.5
5	4	2	5	1	4	4	5	1	5	1	85
6	5	1	4	1	5	1	5	1	4	1	95
7	5	1	5	1	5	1	5	1	5	1	100
8	4	1	5	1	5	2	5	1	5	1	95
9	4	2	4	1	4	2	4	2	4	2	77.5
10	5	1	5	1	5	1	5	1	5	1	100
11	5	1	3	5	5	3	4	2	3	3	65
Mean Score: 86.82											

Table 1. System Usability Scale Scores

### Counterargument Quality Evaluation

To evaluate the quality of the counterarguments, the LLM-based automatic evaluation used by Hu et al. (2023) was utilized and tailored to suit the purposes of the study. Specifically, the study leveraged GPT-3.5 through ChatGPT to evaluate quality

by scoring each evaluation criterion on a scale of 1 to 5, with 5 signifying that the LLM strongly agrees that the counterargument satisfies the criterion. To reduce randomness, each counterargument was evaluated 5 times and their scores were averaged.

The prompt used for evaluation was also based on the study by Hu et al. (2023). It is designed with specific task instructions and a comprehensive list of detailed criteria. The criteria focus on clarity, relevance, validity of reasoning, logical consistency, and overall effectiveness. Details of the prompt with the evaluation criteria are shown in Figure 26. For the sample, a total of 33 counterarguments were used. These counterarguments are the ones generated by the respondents who assessed the usability of the application. There were 11 respondents, and during the application testing, each respondent generated 3 counterarguments. Thus, the total of 33 samples.

The results of the counterargument quality evaluation are shown in Table 2. Out of all the criteria, the "relevance" criterion obtained the highest mean score of 4.98. This means that compared to other criteria, the counterarguments perform the best in terms of directly addressing the proposition and staying focused on the topic. On the other hand, the lowest mean score was 4.55, which was obtained by the "validity of reasoning" criterion. This means that compared to other criteria, the counterarguments performed the worst when it comes to presenting a logical flow of ideas.

Criteria	Mean Score
Clarity	4.65
Relevance	4.98
Validity of Reasoning	4.55
Logical Consistency	4.79
Overall Effectiveness	4.67

Table 2. Counterargument Quality Evaluation Scores

Overall, given the "overall effectiveness" mean score of 4.67, the counterarguments

Proposition: "<input>"

Counterargument: "<output body> \nTo summarize, <output summary>"

You are a lecturer of the writing class. You are given the above proposition and the counterargument that tries to refute it. You need to carefully read the proposition and the counterargument, and evaluate the counterargument based on the criteria:

1. Clarity: The counterargument should be expressed clearly, with a well-defined structure that is easy to follow. Ambiguity or vagueness can detract from the argument's coherence;
2. Relevance: The counterargument should directly address the proposition and stay focused on the topic. Irrelevant points or anecdotal evidence can detract from its coherence;
3. Validity of reasoning: Evaluate the clarity and coherence of the counterargument's reasoning. Is the line of reasoning easy to follow? Does it present a clear cause-and-effect relationship or logical progression? A well-structured and coherent counterargument should present a logical flow of ideas;
4. Logical consistency: Assess the counterargument for internal consistency. It should not contain any contradictory statements or logical fallacies that undermine its coherence. Look for logical connections and coherence between the counterargument's claims, evidence, and reasoning;
5. Overall effectiveness: The counterargument should effectively challenge the initial proposition and provide a convincing alternative viewpoint, and is likely to persuade the reader to reconsider their initial position.

Now you need to assign a score for each criterion on a scale of 1 to 5 where:

- 1 means you strongly disagree that the counterargument satisfies the criterion;
- 2 means you disagree that the counterargument satisfies the criterion;
- 3 means you prefer to be neutral on whether the counterargument satisfies the criterion;
- 4 means you agree that the counterargument satisfies the criterion; and
- 5 means you strongly agree that the counterargument satisfies the criterion.

Note that you should be very strict when giving the score.

Figure 26. Prompt for LLM-based Automatic Evaluation

effectively challenge the initial proposition, provide a convincing alternative viewpoint, and are likely to persuade the readers to consider their initial position. Using the five-point Likert scale scoring range, the mean score for every other criterion is also within the range that represents "Strongly Agree," which is 4.21 - 5.00. This means that the LLM used as an evaluator strongly agreed that the counterarguments generated by the application are clear, relevant, valid reasoning-wise, and logically consistent.

## **CONCLUSION AND FUTURE WORK**

### **Conclusion**

The study was able to develop ArguSure. ArguSure is a counterargument generator Chrome extension designed to provide a way for users to conveniently seek and explore counter viewpoints, especially the ones refuting what they see in digital spaces. Its counterargument generation is powered by Google's multimodal LLM called Gemini. By feeding it the carefully crafted prompts, Gemini, through its API, assesses the input and tries to generate counterarguments for it.

Before using the application, users are required to create and sign in to their account either by providing their basic information or by using their Google account. The application enables counterargument generation using the context menu. This means that if users want to explore arguments against a text they read while browsing, they can just select/highlight it, open the context menu via right-click, and then select the extension option. Aside from just reading the counterarguments, users can also like/dislike and save them. Users can also create topics to further organize the counterarguments they saved. ArguSure also has a web application for users to view and organize the counterarguments they generate. The web application also enables admins to check the application statistics and view or remove registered users.

The system usability of the application and the quality of the counterarguments were tested and evaluated. For the system usability, the System Usability Scale (SUS) was used in which a mean score of 86.82 was obtained. This score corresponds to an "A+" grade rating, indicating an above-average in terms of user experience. For the counterargument quality, LLM-based automatic evaluation was used. The results show that the counterarguments are strongest when it comes to relevance but weakest when it comes to the validity of reasoning, at least compared to each criterion used. The mean

scores obtained for all criteria ranged from 4.55 to 4.98. This can be interpreted as the LLM evaluator strongly agreeing that the counterarguments are clear, relevant, valid reasoning-wise, logically consistent, and effective overall.

### **Future Work**

For future works, using better LLM models or families could further improve the counterargument generation and evaluation. For instance, using Gemini 1.5 Pro or Gemini Ultra instead of the free Gemini 1.0 Pro and GPT-4 or GPT-4o instead of the free GPT-3.5 could elicit more desirable and accurate responses. Aside from changing the LLM model or family itself, experimenting with different model parameter values and prompt designs might also improve the application output. The model parameter values used are just the default values. The prompts used for counterargument generation and evaluation were crafted using different prompt design strategies. However, they can still be improved. A more sophisticated, exhaustive, and experimented prompt can reduce hallucinations and elicit a more desirable response from the LLM.

Additionally, to improve the counterargument evaluation, it is recommended to incorporate humans as evaluators as well. Insights from humans would prove beneficial in determining the quality of the counterarguments generated by the application. Adding human evaluation can make the study's claim regarding the counterargument quality more accurate and legitimate. Related to this, the study can be used further to determine how LLMs compare to humans when it comes to evaluating counterarguments. Comparing human evaluation and LLM-based automatic evaluation would provide additional insights into LLM's potential as an alternative to human evaluators.

With regards to the functionalities of the application, several features could be incorporated into the Chrome extension and its web application for future improvements. The options users can do to the generated counterarguments are limited to like, dislike, and save. For now, this is enough for the purpose of the study. To further help users in



organizing, exploring, and assessing different ideas, however, additional features such as annotations and feedback to the counterarguments could be added. Allowing feedback on the counterarguments can also help not just the users, but also the researchers to know what they could do to improve the counterarguments the application generates. Lastly, ArguSure is currently a browser extension developed only for the Google Chrome browser. For more inclusivity, it is certainly better in the future to extend it to other browsers such as Mozilla Firefox and Apple Safari.

## LITERATURE CITED

- AREEB, Q. M., NADEEM, M., SOHAIL, S. S., IMAM, R., DOCTOR, F., HIMEUR, Y., ... AMIRA, A. (2023). *Filter bubbles in recommender systems: Fact or fallacy – a systematic review*.
- BRUNS, A. (2019, November). Filter bubble. *Internet Policy Review*, 8. Retrieved from <https://policyreview.info/concepts/filter-bubble>
- CARRASCO-FARRE, C. (2024). *Large language models are as persuasive as humans, but how? about the cognitive effort and moral-emotional language of llm arguments*.
- CHAPAGAIN, S. (2022, February). *How to create your own google chrome extension*. Retrieved from <https://www.freecodecamp.org/news/building-chrome-extension/>
- CHEN, B., ZHANG, Z., LANGRENÉ, N., & ZHU, S. (2023). *Unleashing the potential of prompt engineering in large language models: a comprehensive review*.
- CHEN, S. (2023). How social media can solve the problem of “filter bubbles” under the newmedia algorithm recommendation mechanism the example of tik tok. In *Proceedings of the 2023 2nd international conference on social sciences and humanities and arts (ssha 2023)* (p. 1284-1288). Atlantis Press. Retrieved from [https://doi.org/10.2991/978-2-38476-062-6\\_165](https://doi.org/10.2991/978-2-38476-062-6_165) doi: 10.2991/978-2-38476-062-6\_165
- CHEN, S., KHASHABI, D., CALLISON-BURCH, C., & ROTH, D. (2019, July). PerspectroScope: A window to the world of diverse perspectives. In *Proceedings of the 57th annual meeting of the association for computational linguistics: System demonstrations* (pp. 129–134). Florence, Italy: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/P19-3022> doi: 10.18653/v1/P19-3022
- CHIANG, C.-H., & YI LEE, H. (2023). *Can large language models be an alternative to human evaluations?*
- DAXENBERGER, J., SCHILLER, B., STAHLHUT, C., KAISER, E., & GUREVYCH, I. (2020, Jul 01). Argumentext: Argument classification and clustering in a generalized search scenario. *Datenbank-Spektrum*, 20(2), 115-121. Retrieved from <https://doi.org/10.1007/s13222-020-00347-7> doi: 10.1007/s13222-020-00347-7
- FEUERRIEGEL, S., HARTMANN, J., JANIESCH, C., & ZSCHECH, P. (2023, September).

- Generative ai. *Business amp; Information Systems Engineering*, 66(1), 111–126. Retrieved from <http://dx.doi.org/10.1007/s12599-023-00834-7> doi: 10.1007/s12599-023-00834-7
- GARRIDO-MERCHAN, E. C. (2023). *Best uses of chatgpt and generative ai for computer science research*.
- Google ai for developers. (n.d.). Retrieved from <https://ai.google.dev/gemini-api/docs/models/gemini>
- HU, Z., CHAN, H. P., & YIN, Y. (2023). *Americano: Argument generation with discourse-driven decomposition and agent interaction*.
- KANURI, V., CHEN, Y., & SRIDHAR, S. (2018, 11). Scheduling content on social media: Theory, evidence, and application. *Journal of Marketing*, 86, 89-108. doi: 10.1177/0022242918805411
- LEWIS, J. (2018, 03). The system usability scale: Past, present, and future. *International Journal of Human-Computer Interaction*, 1-14. doi: 10.1080/10447318.2018.1455307
- LIU, P., YUAN, W., FU, J., JIANG, Z., HAYASHI, H., & NEUBIG, G. (2021). *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing*.
- MAJER, L., & ŠNAJDER, J. (2024). *Claim check-worthiness detection: How well do llms grasp annotation guidelines?*
- ORBACH, M., BILU, Y., TOLEDO, A., LAHAV, D., JACOVI, M., AHARONOV, R., & SLONIM, N. (2020, July). Out of the echo chamber: Detecting countering debate speeches. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7073–7086). Online: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/2020.acl-main.633> doi: 10.18653/v1/2020.acl-main.633
- PARISER, E. (2018, December). *Algorithmic catastrophe: How news feeds reprogram your mind and habits*. Retrieved from <https://bigthink.com/the-present/facebook-algorithm-filter-bubble/>
- SAHOO, P., SINGH, A. K., SAHA, S., JAIN, V., MONDAL, S., & CHADHA, A. (2024). *A systematic survey of prompt engineering in large language models: Techniques and applications*.

- SILKSTONE, D. (2023). *react-tailwind-chrome-extension-template*. GitHub. Retrieved from <https://github.com/dougwithseismic/react-tailwind-chrome-extension-template>
- STAB, C., DAXENBERGER, J., STAHLHUT, C., MILLER, T., SCHILLER, B., TAUCHMANN, C., ... GUREVYCH, I. (2018, June). ArgumenText: Searching for arguments in heterogeneous sources. In *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Demonstrations* (pp. 21–25). New Orleans, Louisiana: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/N18-5005> doi: 10.18653/v1/N18-5005
- TEAM, G., ET AL. (2024). *Gemini: A family of highly capable multimodal models*.
- VERMA, P., JAIDKA, K., & CHURINA, S. (2024). *Auditing counterfire: Evaluating advanced counterargument generation with evidence and style*.
- WACHSMUTH, H., POTTHAST, M., AL-KHATIB, K., AJJOUR, Y., PUSCHMANN, J., QU, J., ... STEIN, B. (2017, September). Building an argument search engine for the web. In *Proceedings of the 4th workshop on argument mining* (pp. 49–59). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved from <https://aclanthology.org/W17-5106> doi: 10.18653/v1/W17-5106
- WANG, L., CHEN, X., WANG, C., XU, L., SHADIEV, R., & LI, Y. (2024). Chatgpt’s capabilities in providing feedback on undergraduate students’ argumentation: A case study. *Thinking Skills and Creativity*, 51, 101440. Retrieved from <https://www.sciencedirect.com/science/article/pii/S1871187123002080> doi: <https://doi.org/10.1016/j.tsc.2023.101440>
- WANG, M., HU, Y., WU, S., LI, W., BAI, Q., & RUPAR, V. (2024). *Balancing information perception with yin-yang: Agent-based information neutrality model for recommendation systems*.
- WHITE, J., FU, Q., HAYS, S., SANDBORN, M., OLEA, C., GILBERT, H., ... SCHMIDT, D. C. (2023). *A prompt pattern catalog to enhance prompt engineering with chatgpt*.
- YAO, J.-Y., NING, K.-P., LIU, Z.-H., NING, M.-N., & YUAN, L. (2023). *Llm lies: Hallucinations are not bugs, but features as adversarial examples*.