CS410 - Text Information Systems

Final Project Report
Intelligent Browser - Website Topic Summarizer

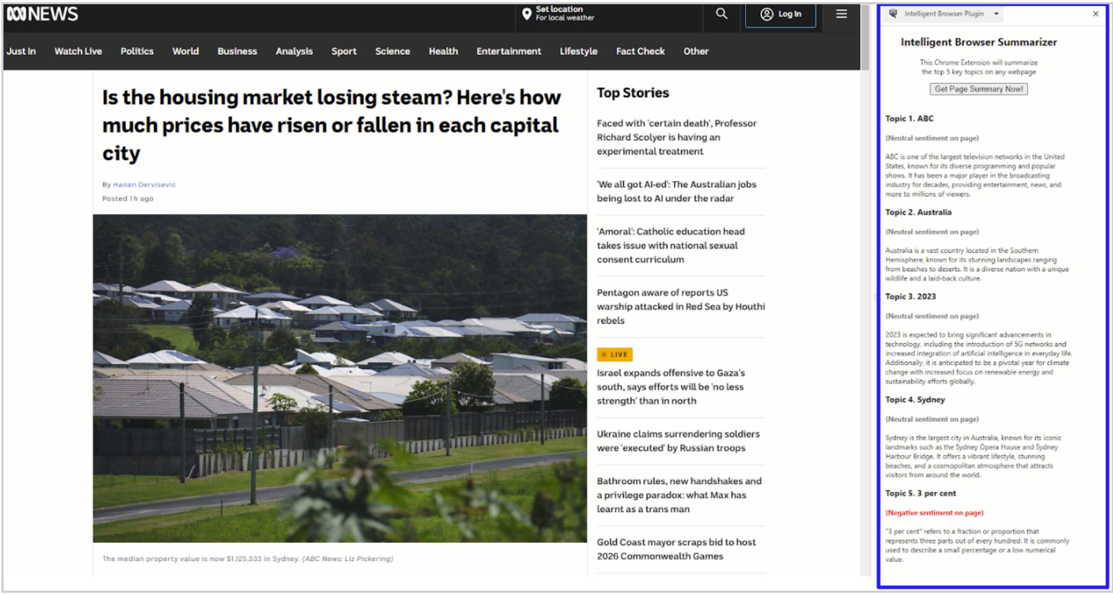
Team: TextRulez

Team Members:

- Tim Crosling (tgc3)
- Avi Nayak (anayak5)
- Srishti Sharma (srishti9)
- Deepthi Abraham (deepthi7)

Abstract:

When reading webpages it is important to identify the key topics first. We have successfully delivered on our proposal to create a tool which allows users to identify the top 5 topics from a web-page, provides a sentiment analysis based on the webpage text, and provides a short summary of that topic. This tool leverages a Chrome extension and a Flask python backend to capture the text of a webpage. The tool then employs a combination of SpaCy Named Entity Recognition (NER) and ChatGPT summarization and sentiment analysis.



Is the housing market losing steam? Here's how much prices have risen or fallen in each capital city

By Hanan Dervisevic
Posted 1h ago

The median property value is now $1,125,533 in Sydney. *(ABC News: Liz Pickering)*

Top Stories

Faced with 'certain death', Professor Richard Scolyer is having an experimental treatment

'We all got AI-ed': The Australian jobs being lost to AI under the radar

'Amoral': Catholic education head takes issue with national sexual consent curriculum

Pentagon aware of reports US warship attacked in Red Sea by Houthi rebels

Israel expands offensive to Gaza's south, says efforts will be 'no less strength' than in north

Ukraine claims surrendering soldiers were 'executed' by Russian troops

Bathroom rules, new handshakes and a privilege paradox: what Max has learnt as a trans man

Gold Coast mayor scraps bid to host 2026 Commonwealth Games

Intelligent Browser Summarizer

This Chrome Extension will summarize the top 5 key topics on any webpage

Topic 1. ABC

(Neutral sentiment on page)

ABC is one of the largest television networks in the United States, known for its diverse programming and popular shows. It has been a major player in the broadcasting industry for decades, providing entertainment, news, and more to millions of viewers.

Topic 2. Australia

(Neutral sentiment on page)

Australia is a vast country located in the Southern Hemisphere, known for its stunning landscapes ranging from beaches to deserts. It is a diverse nation with a unique wildlife and a laid-back culture.

Topic 3. 2023

(Neutral sentiment on page)

2023 is expected to bring significant advancements in technology, including the introduction of 5G networks and increased integration of artificial intelligence in everyday life. Additionally, it is anticipated to be a pivotal year for climate change with increased focus on renewable energy and sustainability efforts globally.

Topic 4. Sydney

(Neutral sentiment on page)

Sydney is the largest city in Australia, known for its iconic landmarks such as the Sydney Opera House and Sydney Harbour Bridge. It offers a vibrant lifestyle, stunning beaches, and a cosmopolitan atmosphere that attracts visitors from around the world.

Topic 5. 3 per cent

(Negative sentiment on page)

"3 per cent" refers to a fraction or proportion that represents three parts out of every hundred. It is commonly used to describe a small percentage or a low numerical value.

Figure 1. Illustration of the Intelligent Browsing Assistant on a sample webpage

IMPORTANT: OpenAI Private Key for Python Server startup

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

(Note: I will keep this key current until the class is graded after which it will be revoked)

Usage Instructions

There are two steps to run the project:

Step 1: Run the Flask Python server:
(a) Ensure you have the right python libraries:

```
pip install flask spacy numpy openai
```

(b) Go to the ‘Server’ folder

(c) Run the python Server:

```
python textAnalyticsServer.py
```

(d) The server will prompt you to enter the openAI private key (note - it usually asks for this twice)

See page 1 - the key is: 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

(e) Note the Server localhost address and port (should be 127.0.0.1:5000)

Step 2: Load the extender into Chrome:
(a) Open Chrome Browser

(b) Enter chrome://extensions/ in the address bar

(c) Toggle 'Developer Mode' to ON at the top right of the page

(d) Click 'Load Unpacked'

(e) Navigate to the directory containing the project and select the ‘ChromeExtension’ folder

(f) Browse to any webpage and toggle the side panel to the right of the plugin

Select "Intelligent Browser Plugin" from the drop down box at the top of the side panel to load the extension

Once the extension has been loaded and the server is running, simply browse to any non-Chrome website (with a http address) and click the ‘Get Page Summary Now’ button

The tool may take a few seconds for the server to run the analytics script, but will return with results (worst case within 10 seconds).

System Design

The tool employs three main components - comprising the user client (Chrome Extension), Flask server on localhost (python script) and calls to the ChatGPT API for summarization and sentiment analysis:

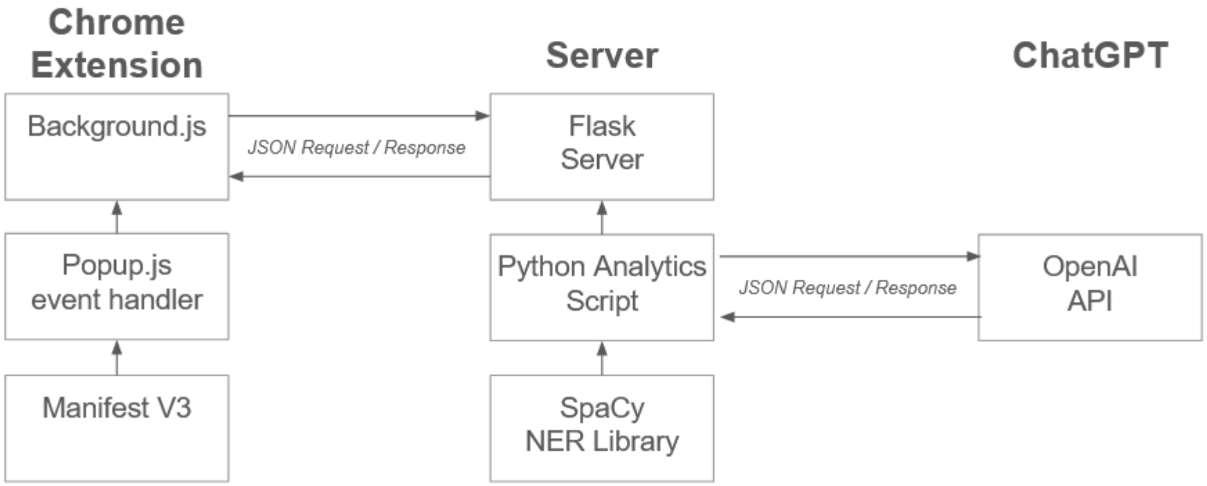


Figure 2. System architecture

The following summarizes the design choices made within each of these components:

Component 1: User Client

- We used a Chrome Extension based on expediency - but also based on initial analysis that showed that while Chrome Extensions are not compatible with Firefox, there is some support for Manifest V3 extensions in Firefox as of version 109. To implement the Chrome extension we built:
  1. **Manifest** - V3 Manifest which links to popup.html for when the extension is clicked and provides an icon image for the extension. The Manifest also supports the popup and the Chrome side-panel. The Manifest then sets up permissions to access localhost port 5000 for the server and runs background.js which implements the JSON Request / Response
  2. **Popup html** - When the user clicks on the chrome extension the popup.html is loaded which provides a base html page for the tool. This includes a button which when pushed triggers the text analysis process, and an empty textCaptured <div> which is used to output the results of the analysis
  3. **Popup.js** - Which registers event listeners for the button being pushed, by which it submits a POST request to background.js using the Chrome message runtime. A second listener is also registered for the response, by which the script will insert the HTML text received by the server into the textCaptured <div> for display.

Component 2: Flask Server

- We leveraged the Flask framework to allow our Chrome Extension to access without violating any Chrome security policies. Flask hosts the textAnalyticsServer.py on localHost port 5000 and supports all request/response functions via JSON
- textAnalyticsServer.py is the main intelligence for the tool Given a url which is passed by the popup.js, the analytics server will execute the following steps:
  a) Capture the innerhtml text content of the webpage using BeautifulSoup. Importantly we decided not to do any data cleansing of the output of BeautifulSoup as the SpaCy and ChatGPT are able to filter this out automatically using IDF ranking
  b) Load the SpaCy core NER model and use it to identify all the topics in the text. We ignored CARDINAL topics on the basis that dates were too abstract
  c) Count the number of occurrences of each topic in the text, and select the top 5
  d) Submit a request to ChatGPT for a short 2 sentence summary of each of the top 5 topics
  e) Submit a second request to ChatGPT for a sentiment assessment for each of the top 5 topics using the entire webpage text string
  f) Format a HTML response incorporating all learnings and returning back to the chrome Extension with a JSON response

Component 3: ChatGPT API

- We used the openai python library to easily connect to the ChatGPT API. We selected the latest gpt-3.5-turbo library and submitted our query string as a user role. The response is then processed
- In the case of the sentiment analysis we simply filtered the ChatGPT response using a positive / negative / neutral categorization

Implementation Considerations

In developing the tool there were a number of key challenges identified. The following provides a summary:

Challenge 1: Topic Identification
Description: Initial plan was to use BM25 to identify top words on the webpage however this proved to be unreliable in identifying true topics within the webpage. BM25 also was unable to support multi-word topic labels which ultimately impacted the relevance of the topics returned by the tool.

Solution: SpaCy comes with a pre-trained model for NER topic identification. We found this model to be extremely efficient and accurate. We further refined the topic identification by ranking the topics identified by SpaCy according to term frequency in the webpage text.

Challenge 2: Very slow runtime for sentiment analysis and summarization
Description: Experimentation with BERT for sentiment analysis proved to be complex in training the model - and delivered very slow response times which were not practical for a web app. The model was also sensitive to the training data used - e.g., we tried testing with newspaper content but found that this was less useful for non-editorial webpages.

Solution: ChatGPT provides a highly efficient sentiment analysis and summarization engine which we found was more accurate / robust. Frankly, ChatGPT was so good we felt it was important to use in place of our own draft.

Challenge 3: Integration between Javascript and Python analytics libraries
Description: The Chrome Extension is written in javascript, but most text analytics libraries are available in python. We looked at javascript library ports (e.g., SpaCy-js, BERT-js) but felt that they were sub-capable vs. python. We also looked at PyScript as a workaround but found that Chrome's Manifest V3 security policies limited our ability to call external libraries (and compiling our own local runtime was VERY complex). Finally, we looked at hosting the python library on Google Cloud as a cloud function but found this also to be a significant add to the scope of the project.

Workaround: Flask provides a relatively simple way to host a python script on a localhost server, which can then be integrated into a javascript JSON request/response flow. While not ideal, we felt this was a reasonable compromise to allow us to focus on the text analytics options.

Challenge 4: Gitlab security policy rejection of OpenAI private key
Description: A private key is needed to access the ChatGPT API; however GitHub security policies don't allow private keys to be hosted on the repository. We looked at alternate ways to host the key including encrypting the key in a text file and using a passphrase to access, but were not able to circumvent the security policy.

Workaround: Rather than diving deep into crypto solutions we felt this was a distraction and so simply created a text prompt to allow the API key to be manually entered when the Flask python server is started. The key can be found on page 1 of this report.

Challenge 5: ChatGPT input limited to 4096 tokens
Description: For very large websites we encountered a bug where the server will crash if the ChatGPT query exceeds 4096 tokens. The code works fine for smaller websites (the vast majority) - but for large websites this is a problem

Workaround: We ran out of time to solve this problem; which would require chunking up the website corpus into 'batches' of text no greater than 4096 words.

Conclusion

Our tool delivers a utility and is relatively easy to use. The text analysis is insightful - particularly when you scroll down to the fourth or fifth topic. There are ample areas for improvement including: Streamlining the chrome extension install process, migrating the python script to a Google function, adding URL links to each topic on WikiPedia for a double-click.

While this project was challenging we learned an enormous amount about this space - and built a functioning tool!. The following were our key learnings:

(1) **Readily available text mining and analytics open source** - there are MANY pre-trained models out there that can be used quickly. SpaCy is REALLY impressive both in ease of use, speed and accuracy. We could not replicate the accuracy this library provides out of the box with 3 lines of code

(2) **ChatGPT is AMAZING.** We are so lucky to be able to access world leading innovation with a simple search string for free (up to a limited number of queries). ChatGPT also helped when we got stuck by providing coding guidance (e.g,. Providing details on how to implement the Flask python server)

(3) **Building this tool required a cross-functional skillset** including frontend javascript, backend server, python text analytics and cryptographic security work for the openAI key. This reinforces the value of learning content across multiple domains.

Sources

1. How to create a Chrome Extension in 10 minutes flat (https://www.sitepoint.com/create-chrome-extension-10-minutes-flat/#:~:text=What%20Is%20a%20Google%20Chrome%20Extension%3F%201%20Step,Styling%20...%205%20Step%205%3A%20Test%20the%20Extension)
2. Welcome to Manifest V3 - Chrome Developers (https://developer.chrome.com/docs/extensions/mv3/intro/)
3. Learn how to build a Custom Named Entity Recognition (NER) model using SpaCy Youtube video (https://www.bing.com/videos/riverview/relatedvideo?&q=spacy+ner&&mid=1B229B4EDF26394F3FBC1B229B4EDF26394F3FBC&&FORM=VRDGAR)
4. Classify text with BERT (https://www.tensorflow.org/text/tutorials/classify_text_with_bert)
5. OpenAI Python API Library (https://github.com/openai/openai-python)
6. How to host your Flask App on Python anywhere for Free (https://medium.com/swlh/how-to-host-your-flask-app-on-pythonanywhere-for-free-df8486eb6a42#:~:text=How%20to%20Host%20Your%20Flask%20App%20on%20PythonAnywhere,5%20Step%205%3A%20Configuring%20the%20root%20file%20)
7. ChatGPT support for setting up the Flask Server (openAI.com)