

# Software Documentation & Report

## *Glass Trend - An Extension for Analyzing Employee Satisfaction Trend*

Name: Huijun Lin (One person group)

NetID: huijunl2

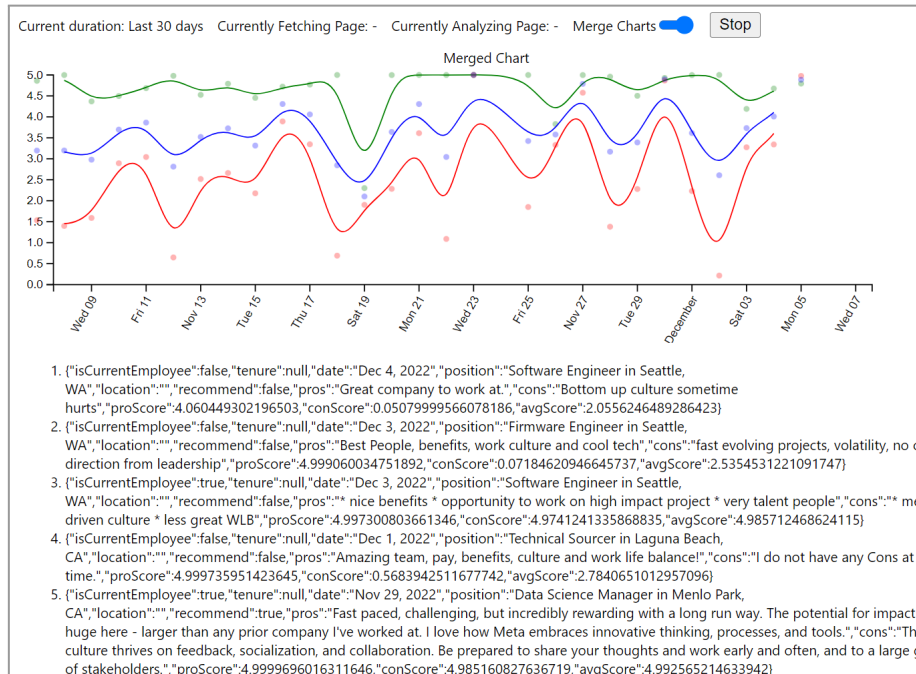
Email: huijunl2@illinois.edu

Subject Areas: Free Topics

## 1. Overview

### 1.1 Goal

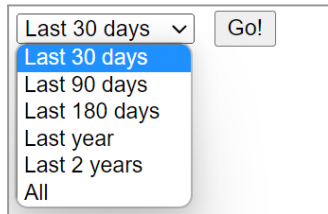
The goal of my project is to implement a chrome extension that provides Glassdoor users a historical review sentiment trend, in order to help them better understand the company they are currently viewing.



1. review sentiment trend for Meta for last 30 days on Dec 7th, 2022

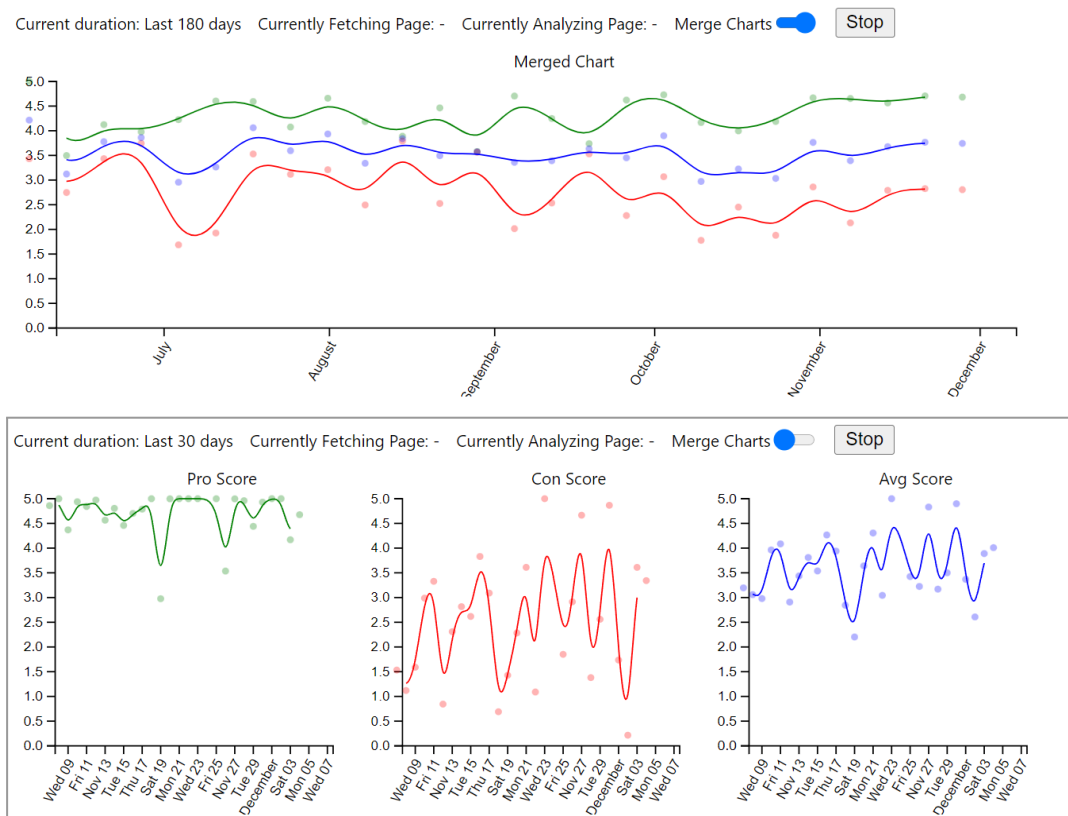
## 1.2 Functions

1. In a Glassdoor company page, a user can preview the historical review trend with a "pros" curve (green), "cons" curve (red) and "avg" curve (blue).
2. Users can change the time range of the target reviews.



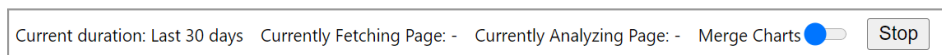
2. time range options for including target reviews

3. Users can switch between merged or split when viewing the "pros", "cons" & "avg" curves.



3. split view of the trends

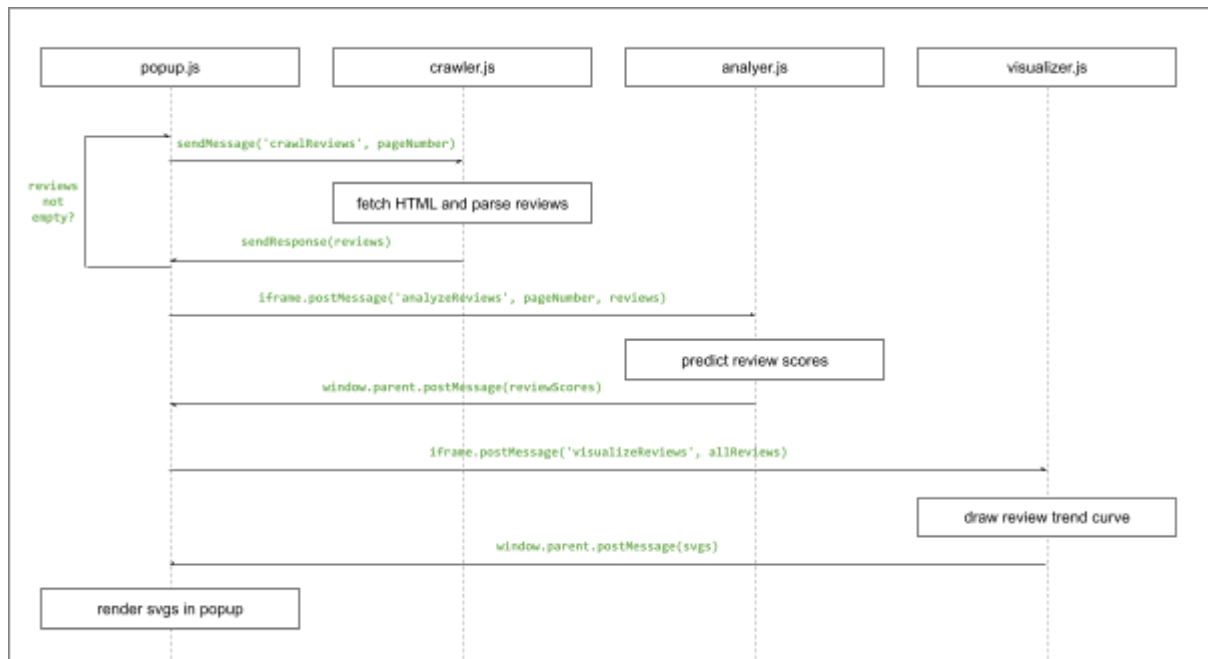
4. Users can stop fetching incrementally at any point and the extension will perform based on the gathered information.



4. button to stop fetching

## 1.3 Implementation Brief

The project was broken down into 4 components - popup.js, crawler.js, analyzer.js and visualizer.js:



- **crawler.js**: Given the current Glassdoor url and a page number, the crawler component fetches the target paginated html document, extracts the review contents and returns them in the format of a JSON object list.
- **analyzer.js**: Given the review content JSON list responded by the crawler component, the analyzer component returns sentiment analysis scores based on a CNN model by using TensorFlow.js.
- **visualizer.js**: Given the sentiment analysis scores for each review, plots the data points with x-axis being the date of the review and y-axis being the score of the review.
- **popup.js**: Renders the user control panel within the extension popup, as well as the curves of the review trend and the review list.

## 2. Implementation Details

This section explains each component accordingly and lists the worth noting technical details, in the order of implementation steps.

### 2.1. Crawler.js

#### 2.1.1. Accessing the HTML content of the current page.

I was initially getting an error complaining that I was forbidden to access the HTML content of the current page. As suggested by Chrome, scripts that deal with page contents should be configured as **content\_scripts** in the **manifest.json**.

By looking closer into the page content, I was able to extract the "pros" and "cons" reviews and other metadata like review date, by using the css selector. For example, `span[data-test="pros"]` can locate the "pros" reviews.

### 2.1.2. Fetching paginated pages asynchronously.

I realized the pattern of Glassdoor company url is in the format of:

```
> www.glassdoor.com/Reviews/{company_name}-Reviews-{id}_P{page_number}.htm
```

For example:

```
> www.glassdoor.com/Reviews/Meta-Reviews-E40772_P2.htm
```

Since the `id` will not change on pagination, I can modify the `page_number` in order to fetch a paginated page for the given company. After some research, I decided to use the native [fetch API](#).

### 2.1.3. Returning review object list to the main popup script.

The component is only in charge of crawling and its output should be sent back to the main popup script. I do so by making use of the [message passing](#) mechanism provided by the Chrome Extension APIs.

## 2.2. Analyzer.js

### 2.2.1. Exploring TensorFlow.js (tfjs)

Because I am building a chrome extension, without setting up a server running another language, I am limited to building a sentiment analysis module with javascript. That means, most of the popular Python like BERT, NLTK or SpaCy are all not available for me. After some investigation, I decided to try using TensorFlow.js, which is the JavaScript version of TensorFlow.

### 2.2.2. Picking the model

By exploring different models, two models stand out: the [CNN](#) model and the [LSTM](#) model. I did some research online and also experimented with both and eventually went with the CNN model. Because I noticed that the review sentences are usually short or constructed with simple words, which is more suitable for the CNN model.

### 2.2.3. Placing the component inside a Sandbox iframe

When I start importing the tfjs, I notice that it is forbidden with the error:

```
> 'unsafe-eval' is not an allowed source of script
```

I spent a lot of time investigating and it was due to the new manifest v3 of Chrome Extension, which blocks using the eval function. For future references, I want to point out that almost all the current answers on the internet are outdated. The correct fix should be creating a sandbox iframe for the unsafe codes and communicating with the iframe's parent window, i.e. the popup.

### 2.2.4. Sending data to and receiving data from the tfjs API

Since the tfjs code is restricted in the sandbox iframe, only the parent window can communicate with it. And the way to have the two windows talking to each other is through [postMessage](#).

In order to send data from popup to iframe:

```
> document.getElementById('sandbox').contentWindow.postMessage(data, '*');
```

In order to send data from iframe to popup:

```
> window.parent.postMessage(data, '*');
```

### 2.2.5. Iterations on predicting reviews

For each review, I turned each word into a word index in the CNN model. And then generated a fixed-length number sequence representing the review. Eventually, I call the `tf.tensor2d` with the sequence to receive a tensor object with the prediction function.

Initially, I concat the "pros" and "cons" reviews into one paragraph and then get a prediction score with the method above. I noticed that the scores are almost always 0.99+. By debugging I realized if the sequence limit was set to 100 words by default and a lot of the times the "pros" review has more than 100 words. I made two changes accordingly to address the issue:

1. change the sequence limit from 100 words to 1,000 words.
2. evaluate the "pros" and "cons" review separately, use the average prediction score of the two as the final score.

By doing that, I observed there is actually value in showing the "pros", "cons" and the average scores together. For example, although the average score seems flat, the pro score and the con score are not necessarily flat.

## 2.3. Visualizer.js

### 2.3.1. Exploring D3.js

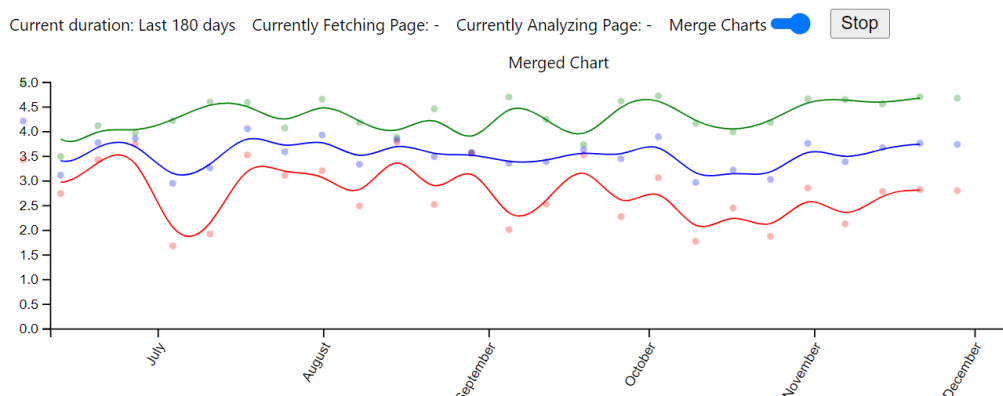
I explored what are the available JavaScript libraries for drawing charts. I eventually decided to use [D3.js](#) because the community is big and a lot of related questions are available in StackOverflow.

### 2.3.2. Plotting the data points and drawing the curves

After I transformed the reviews into an array of data points, I realized they are too many for humans to draw conclusions from and too many for D3.js to draw a curve that can represent the trend.

Therefore, I group the reviews by date and use an average score on each date. I found it working at first, but when I was fetching the reviews older than 180 days ago, the old problem occurred again.

I eventually decided to use different data granularities, for date range equal or more than 180 days, I group the reviews by the start of the week for each date. This treatment turns out to be working well.



## 2.4. Popup.js

### 2.4.1. Connecting all components

For the scripts outside of the sandbox iframe, I relied on the message passing mechanism provided by the Chrome Extension APIs:

```
> chrome.runtime.onMessage.addListener()
```

For the sandbox iframe, I relied on the native method `postMessage` for message passing.

### 2.4.2. Polishing the UI

I added more messages to let users know what is currently happening behind the screen, such as the "currently fetching page" and "currently analyzing page".

Also, I added a "merge charts" button to allow users to switch between the merged chart and the split charts. There is also a "stop" button to stop the extension from fetching further pages.

### 2.4.3. Adding review object list for logging & debugging

During implementation I encountered a few bugs. In order to debug more efficiently, I printed the review object list below the chart. I also found it useful when I want to understand reviews that happened on a specific date.








## 3. Installation and Usage

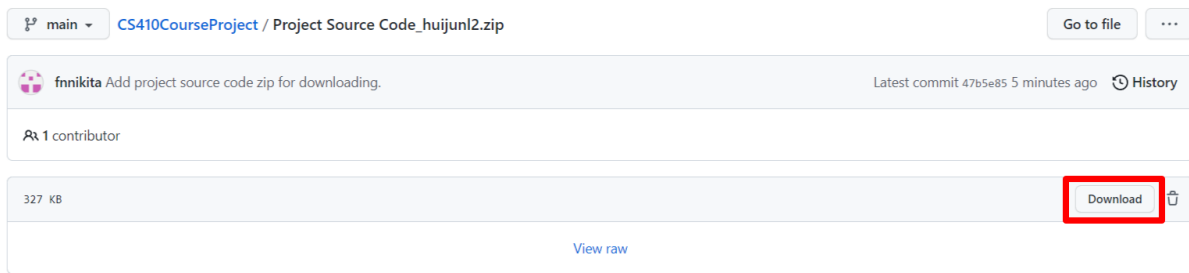
Before installing the extension, you should have a chrome browser. If not, you should download and install it first.

Then open your chrome browser.

### 3.1. Download the source code

3.1.1. The Chrome extension is not yet in the web store. The source code file is in my Github's CS410CourseProject repository. Since it is a file, I suggest you directly download the zip file of it, you can download from [here](#). Remember the local location you save the zip file.

	fnnikita Add project source code zip for downloading.	47b5e85 4 minutes ago	 6 commits
	Project Source Code_huijunl2	Add project source code.	38 minutes ago
	Project Progress Report _ huijunl2.pdf	Add progress report	25 days ago
	Project Proposal _ huijunl2.pdf	Project Proposal	2 months ago
	<b>Project Source Code_huijunl2.zip</b>	Add project source code zip for downloading.	4 minutes ago
	README.md	Update README.md	2 years ago

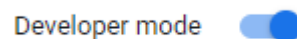


3.1.2. Decompress the zipped folder, and move it somewhere permanent. Note that deleting or moving the decompressed extension folder after completing these steps will result in the extension not working in the browser.

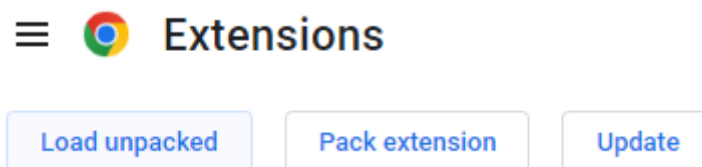
## 3.2. Install the extension

3.2.1. In your Chrome browser, paste `chrome://extensions/` into the address bar and navigate to the page.

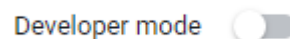
3.2.2 Toggle on "Developer Mode" (at the top right of the screen).



3.2.3. Click "Load Unpacked", and select the decompressed extension folder.



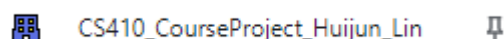
3.2.4. Toggle off "Developer Mode"



3.2.5. Click the "Extensions" icon in the Chrome toolbar (puzzle at top right of toolbar).



3.2.6. For convenience, Pin the extension.



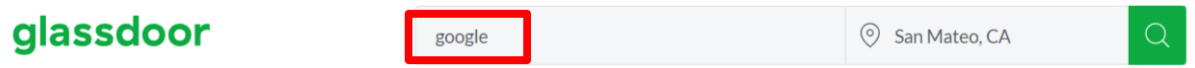
### 3.3. Open a company's glassdoor review page

3.3.1. Go to the homepage of glassdoor. You can sign in using your own glassdoor account. Or you can use this testing account to log in . Please note that the account is only for testing the extension, please do not change the password.

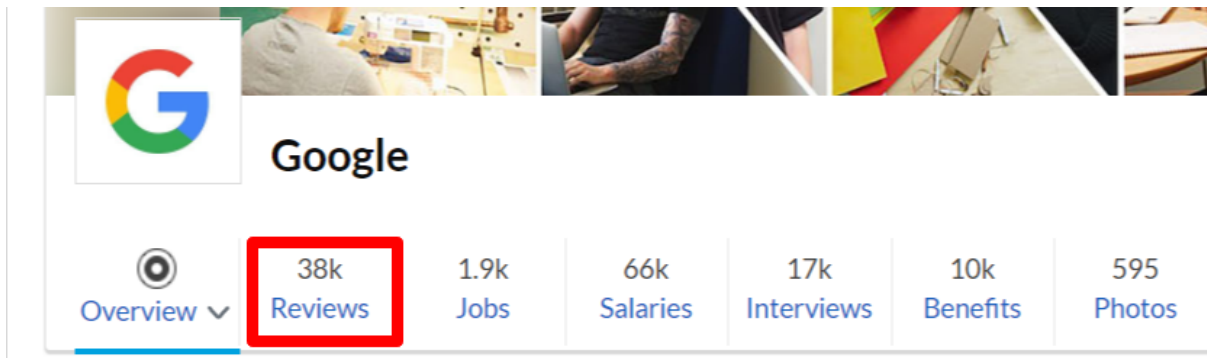
Testing account: glassdoor2023test@gamil.com

password: glassdoortest2023

3.3.2 Go to a company's page on Glassdoor, for example, google.



3.3.3 Click the reviews tab on the company's page.

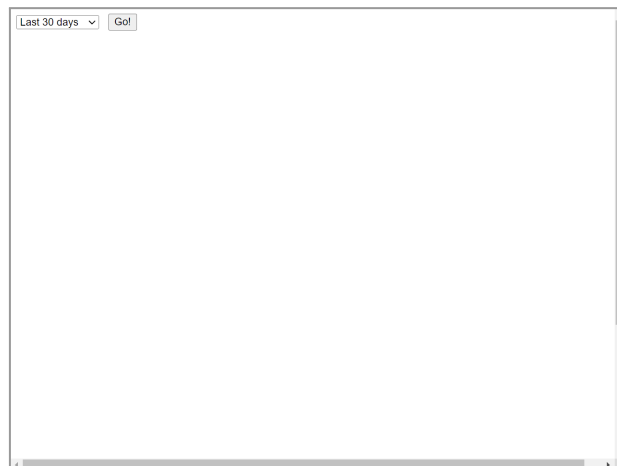


### 3.4. Open the extension popup

3.4.1 Open the popup by clicking the extension icon.

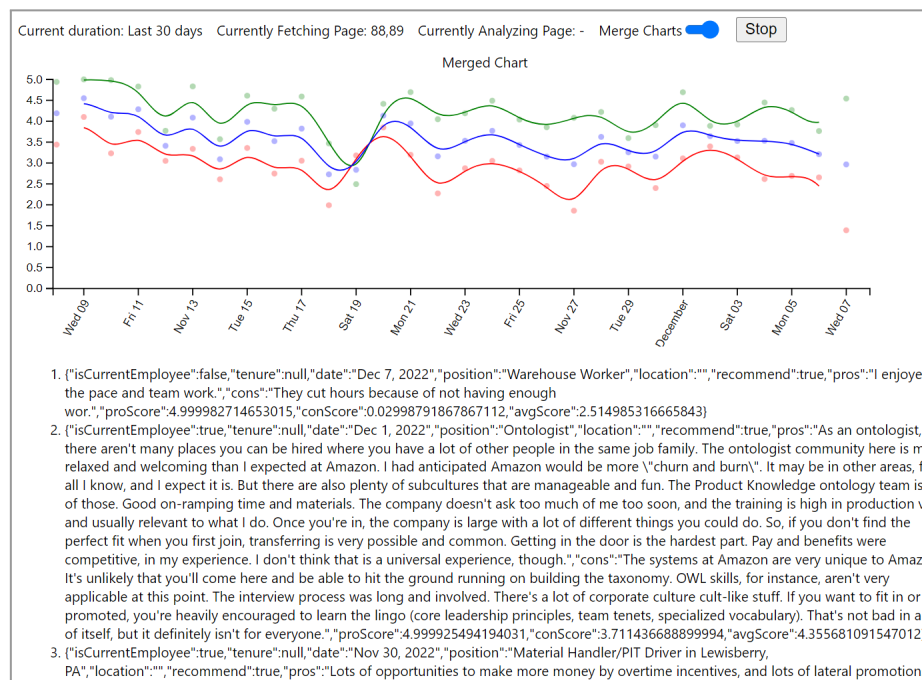


It should say "loading model..." for the first few seconds, and then the popup should show a date range selection and a "Go" button.

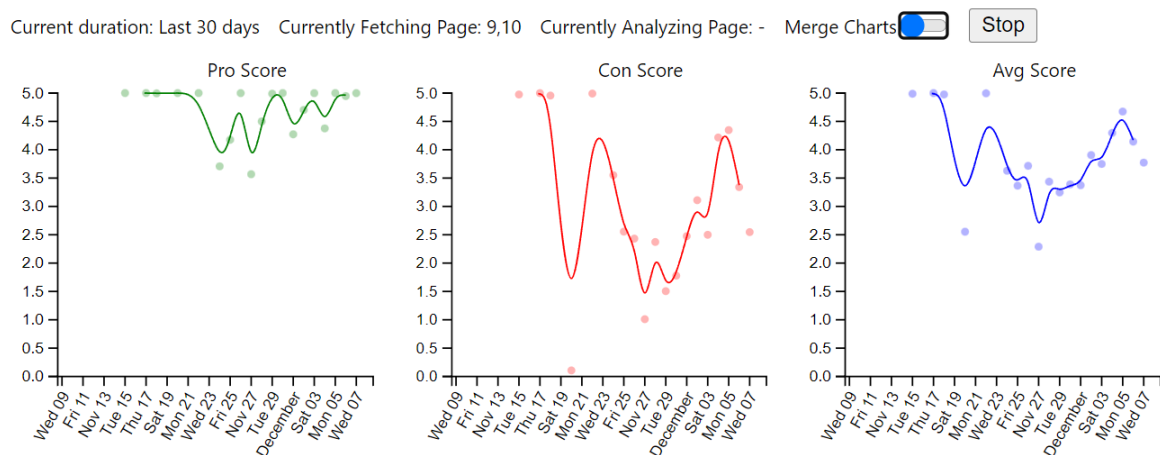




3.4.2 Choose a time period and click the "Go!" button . Wait a few seconds, then you can observe the results. Note that the green line represents positive comments, the red line represents negative ones, and the blue line represents the average score of the two.



3.4.3 If you drag the "Merge Charts" button from right to left, you can see the separate charts, which may help you better understand which part causes the average trend changes.



Note that the curves do not connect the starting and ending data points because I am using "open" curves to connect the data points. I've evaluated both basic and open curves and decided to use the latter.

## 4. Self-evaluation

### 4.1. Project Scope

I have completed the project's requirement & expectation I listed in the project proposal:

#### 4.1.1. The extension should be able to run for different company pages in Glassdoor.

I have experimented with the extension with multiple companies in Glassdoor, such as [Google](#), [Meta](#) and [Walmart](#). And the extension is working as intended.

#### 4.1.2. The extension should be able to plot each data point with the coordinate (time, score).

I have implemented this feature and also extended it by adding estimated curves.

### 4.2. Evaluation

#### 4.2.1. Test different companies and display different results (assuming none of the companies will have the same result).

This is validated. Indeed each company has a different curve.

#### 4.2.2. Evaluate the Crawler component by checking the targeted reviews are fetched.

I have manually tested this, the extension is able to fetch all reviews given a target url.

#### 4.2.3. Evaluate the Analyzer component by checking whether the returned score for a given review is reasonable (involves human effort). Evaluate both positive & negative reviews.

This is working as intended. As I mentioned in the implementation details section, from the beginning, I kept getting scores close to 0.99+. And I've debugged and fixed the issue by predicting the pros and cons review score separately and then using the average score as the final score. After that, I manually checked the prediction score for each review and I found them reasonable.

#### 4.2.4. Evaluate the Visualizer component by checking whether the rendered time chart matches the reviews.

I have validated by monitoring the rendered chart with only the first 10 reviews using the browser inspector. The visualizer is drawing the data points and curves correctly.

## 5. Team Member Contributions

Since it's a one-person group, I did the whole project independently. I used 40 hours in this project.