Comquest: Large Scale User Comment Crawling and Integration

Zhijia Chen
Department of Computer & Information Sciences
Temple University
Philadelphia, USA
zhijia.chen@temple.edu

Lihong He
IBM Almaden Research Center
San Jose, USA
lihong.he@ibm.com

Arjun Mukherjee
Department of Computer Science
University of Houston
Houston, USA
amukher6@central.uh.edu

Eduard Dragut
Department of Computer & Information Sciences
Temple University
Philadelphia, USA
edragut@temple.edu

ABSTRACT

User-generated content like comments are valuable sources for various downstream applications. However, access to user comments data is often limited to specific platforms or outlets, which imposes a great limitation on the available data, and may not provide a representative sample of opinions from a diverse population on a particular event. This paper presents a comment crawling system that leverages the Web API of popular third-party commenting systems to collect comments from a large number of websites integrated with the commenting systems. Given a target page, the crawling system utilizes a deep learning model to extract API parameters and send HTTP requests to the API to retrieve comments. The system, Comquest , that we propose to demo is news-oriented and crawls comments regarding specific news topics/stories. Comquest can work with any website that allows commenting. Comquest provides a useful tool for collecting comments that represent a wider range of opinions, stances, and sentiments from websites on a global scale.

CCS CONCEPTS

• **Information systems** → **Deep web**; **Web crawling**; *Information integration*; *Information systems applications.*

KEYWORDS

comments, crawling, Web API

ACM Reference Format:
Zhijia Chen, Lihong He, Arjun Mukherjee, and Eduard Dragut. 2024. Comquest: Large Scale User Comment Crawling and Integration. In *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24), June 9–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3626246.3654736

SIGMOD-Companion '24, June 9–15, 2024, Santiago, AA, Chile

ACM ISBN 979-8-4007-0422-2/24/06…$15.00
https://doi.org/10.1145/3626246.3654736

1 INTRODUCTION

Web 2.0 technologies have significantly transformed the Internet landscape by enabling individuals to participate in content creation. Commenting features have gained widespread acceptance by online media, with their audiences becoming active contributors of content published on their websites. Social scientists argue that commenting features increase user-to-user interactions and contribute to shaping a democratically valuable and vivid interpersonal discourse on topics of public interest. User comments fuel a wide range of applications, such as opinion mining [8, 11], fake news detection [7] and, potentially, training large language models [3], which attract relentless attention from industry and academia alike.

Research on user comments is increasingly important for understanding public opinion and sentiment regarding social events [1, 9]. However, data and social scientists face limited access to user comments from diverse sources. They generally rely on existing user comments datasets that are collected from a limited number of sources (websites). Such approach may lead to biased or incomplete datasets, as different news outlets or social media platforms may attract users with certain demographic characteristics, interests, and ideologies (e.g., most users commenting at New York Times are progressive compared to those at Fox News who are conservative) [10]. Thus, collecting user comments from multiple sources is crucial for obtaining a comprehensive view of public opinion.

Most websites do not create the commenting system themselves. They employ third-party commenting systems, which allow users to post comments transparently on the websites and to items (e.g., an article) of their choice. Under the hood, a webpage is integrated with some JavaScript code that is executed by Web browsers to interact with the commenting system's backend by sending HTTP requests to the commenting system's server via its Web API. According to BuiltWith (a company that tracks Internet technology trends), there are over 1 million websites that have integrated commenting system technology as of 2023[1]. The wide adoption of third-party commenting systems opens the possibility of large-scale multi-source comments crawling with a unified solution. To this end, we present Comquest, an efficient and scalable user comments crawling system capable of collecting comments on news from a wide range of websites at scale. Our user comments crawling system offers the following features:

[1] https://trends.builtwith.com/widgets/comment-system

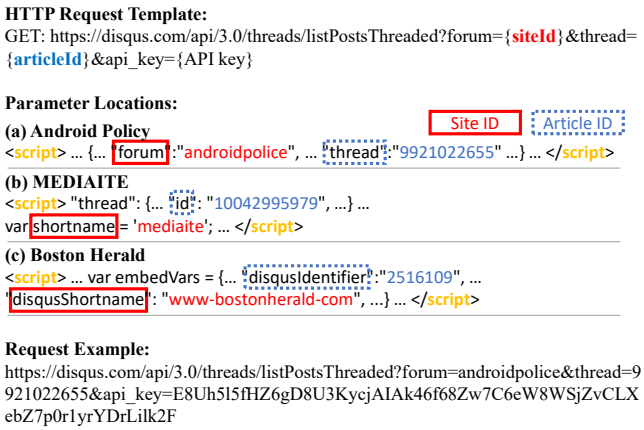

Figure 1: Disqus HTTP request templates and examples. The parameter values in the URL are from HTML documents.

Web API Scraping: Comquest crawls comments via the Web API of three main-stream commenting systems on the market, including Disqus[2], OpenWeb (formerly Spot.IM)[3], and Viafoura[4], targeting a large scale of websites that utilize these commenting systems.
Flexible Crawling: the back-end crawler of Comquest takes the URL of an arbitrary webpage as input, and it tries to retrieve comments if a registered commenting system is detected. While the front-end web portal is hooked to Google News, it also allows the users to set up tasks on any interested set of websites.
Dynamic Tasking: Instead of building a crawler with hard-coded target news or fixed input tasks at the start, we designed a crawler controller that allows users to manage crawling tasks dynamically such as adding/removing target URLs and setting crawling parameters (e.g., news revisit interval and expiration time) to help the crawler to stay efficient and polite.
Horizontal Scaling: The crawler controller is decoupled from the backend crawling service and acts as a centralized task manager. This design enables horizontal scaling, allowing users to spin up more crawler instances to increase throughput and track multiple ongoing stories.

The data collected with Comquest has supported a number of previous works [2, 4, 5]. We have collected hundreds of millions of user comments related to major news events. For instance, Comquest visited 94k news articles and crawled 6.4m comments during the 2022 midterm elections, and it has collected 3m comments w.r.t the Russian-Ukraine war by March 1, 2024.

2 COMMENT RETRIEVAL

In this section, we give an overview of the core method of retrieving comments implemented in Comquest. We will present the whole crawling system in the following section.

Comquest imitates a webpage's interaction with its commenting system to retrieve comments from the server by sending direct HTTP requests. The process requires an HTTP request template and parameter values specific to the target webpage. For example, Figure 1 shows the request template of the Disqus commenting system and a request example. Several parameters are required to communicate with a commenting system, but the typical ones are `siteId` and `articleId`.

The HTTP request template of a commenting system can be used to request comments for all the websites using the commenting system. Thus as a one-time effort, we manually acquire the template from the systems' API documentation or infer the template from the comment-loading HTTP requests issued by a browser to the commenting systems.

The problem of retrieving comments is thus to instantiate the commenting system's HTTP request template with the parameters of a target webpage. We observe that the parameters are generally stored in the source HTML string of the target webpage. This is because comments on a webpage are typically loaded by Web browsers, which requires access to all the parameters based on the HTML source codes.

A naive solution for the parameter extraction problem is to compose rules and regular expressions, which is unscalable because it would need to compose the rule for each website since websites may store the parameters at different locations of HTML and associate them with different names, as illustrated by the three Disqus parameter source snippets in Figure 1, e.g., the `siteId` parameter is associated with 'forum', 'shortname' and 'disqusShortname', respectively. Furthermore, the solution is also sensitive to website style/layout changes.

We approach the parameter value extraction from the *sequential labeling* angle. Specifically, given a set of parameters $\Phi = \{\phi_1, \phi_2, ..., \phi_n\}$ and an HTML string, we aim to assign each character a label that represents a parameter in Φ. For example, we assign '1' to each character of a value of `siteId` and '2' for `articleId`. If the model is not confident about '1' or '2', it will assign `o` to a character, which stands for `other`. The following shows the labeling for the parameter location example (a) in Figure 1:

```
..."forum":"androidpolice"..."thread":"9921022655"...
ooooooooooo11111111111ooooooooooooo2222222222ooooo
```

We apply a deep learning-based solution for character-level labeling based on the BiLSTM model. To train the model, we collect an extensive dataset from the Common Crawl Project. We filter the webpages that show apparent code signals (e.g., loading the commenting system libraries) of the supported commenting systems.

3 SYSTEM DESCRIPTION

In this section, we present the architecture and workflow of Comquest . As depicted in Figure 2, the system comprises three major components: a front-end Web portal for user interaction, a controller implemented with a relational database, and backend comment crawlers. We discuss each component in detail below.

3.1 Crawler Web Portal

The Web portal is implemented with the React and Flask frameworks. The portal provides the following features:
Task Management: The portal helps users to setup and edit crawling tasks. It features a news feeds page dedicated to task creation. As illustrated in Figure 3 (a), the page displays the latest Google News feeds to the user in a hierarchical structure, allowing them

[2]https://disqus.com/
[3]https://www.openweb.com/
[4]https://viafoura.com/

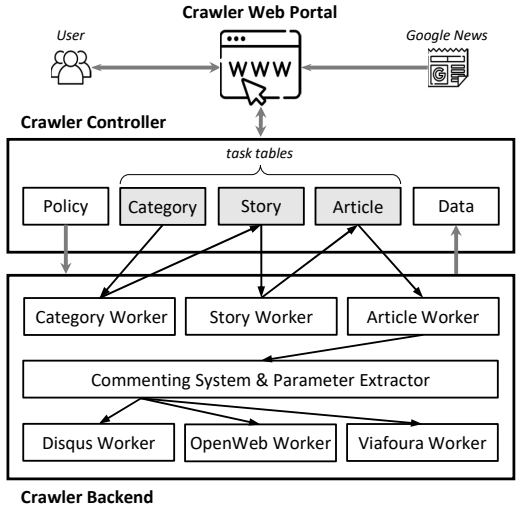

Figure 2: The System Architecture of Comquest .

to create a crawling task targeting news at different granularity. Following the Google News feed, we organize news into three hierarchical levels, including Category, Story, and Article. A news category is a broad grouping of news stories based on their topics or subject matter. A story is a news event, which is composed of multiple articles that provide different perspectives. And finally, an article is a piece of content that provides information or commentary on a particular story. An article is the basic news unit that Comquest works on.

To manage existing tasks, we organize them in an interactive table, as shown in Figure 3 (b). Each row of the table presents an overview of a task, including its status and the number of articles/-comments crawled under the task. Users can click a task field to examine it in detail. Clicking on a task title, for instance, will display its subtasks in a new table view. Figure 3 (c) displays the stories under the "2022 Mid-term Election" task. Clicking on the number of articles/comments will present the distribution of the top 10 outlets for the data, as demonstrated in Figure 3d. Finally, the status field allows users to turn a task on or off.

Crawler Monitor. This feature monitors the latest activity and the contribution history of the crawler. As shown in Figure 4, we use a dynamic line chart to show the number of comments/articles the crawler collected in the last 60 seconds in real-time. The contribution history is presented in an activity calendar that shows the daily number of comments collected from each platform in the past year. This feature may also help as a maintenance tool. If the HTTP request template of a commenting system is broken (e.g., due to the API update), the users can notice its zero activity. On the admin side, Comquest includes an alert mechanism to notify the admin of possible broken HTTP request templates.

Policy Settings: Existing studies have shown that users' interest in commenting a news article peaks within the first 36 hours after it is published. But the commenting activity wanes slowly, and it may last several days with a long-tail distribution [4]. This indicates that revisiting target news with appropriate intervals is essential for the crawler to balance content freshness and politeness. Therefore, we allow users to configure the initial revisit intervals and the expiration time. The crawler measures the arrival rate of comments and adjusts its parameters dynamically. The objective is to achieve optimal freshness under politeness constraints (i.e., the number of requests) [6].

3.2 Crawler Controller

Comquest is designed with scalability and flexibility in mind, requiring the crawler to run as a service that accepts crawling tasks from users and the ability to have multiple instances work together to increase the processing capacity. To this end, we design a control interface that utilizes a relational database to manage tasks, crawler policies, and output data.

The task interface comprises three tables corresponding to the three news structures: category, story, and article. Each row of the task tables contains details of a target news task, such as the URL, task status, and parent task (if applicable). Tasks can be added to the interface via the Web portal or as subtasks of a parent task, such as a story under a category task (see Section 3.3).

The policy table stores all parameters related to the crawling policy, as discussed previously. The data table provides a universal interface to integrate comments from various providers into a unified format. Specifically, the crawler receives comments from different providers in different JSON structures. We treat a comment as a text message associated with common attributes such as the author, posting time, and a reference to the parent message (if it is a reply).

3.3 Crawler Backend

The crawler service comprises independent news workers that track task tables at the three news levels and plugin-specific comment crawlers that crawl comments related to each article. As shown in Figure 2, a category worker queries a task from the category table and crawls the latest stories or articles at each visit. The output stories and articles are added to their corresponding tables to create subtasks under the category. Similarly, a story worker crawls the articles under the story and creates subtasks in the article table. Finally, an article worker tries to detect a registered commenting system in the HTML source for each article page based on the code signals of the commenting system, such as containing the commenting system library or server domain. If a commenting system is detected, the article worker extracts the parameters of the HTTP request template using the deep learning model described in Section 2 and invokes the corresponding comment crawler.

Note that category, story and article workers are running concurrently, and we may start multiple instances of the workers to increase the crawling capacity. The upper limit of the instances is determined by the maximum concurrent connections of the controller database, which is dependent on the hardware, but even a moderate modern PC is capable of handling over 100 concurrent connections. We rely on the row-level locks of the database to avoid multiple workers working on the same task.

4 DEMONSTRATION SCENARIO

We will invite visitors to test Comquest using the Web portal and demonstrate several scenarios to show the advantage of Comquest compared to traditional website-specific crawling methods.

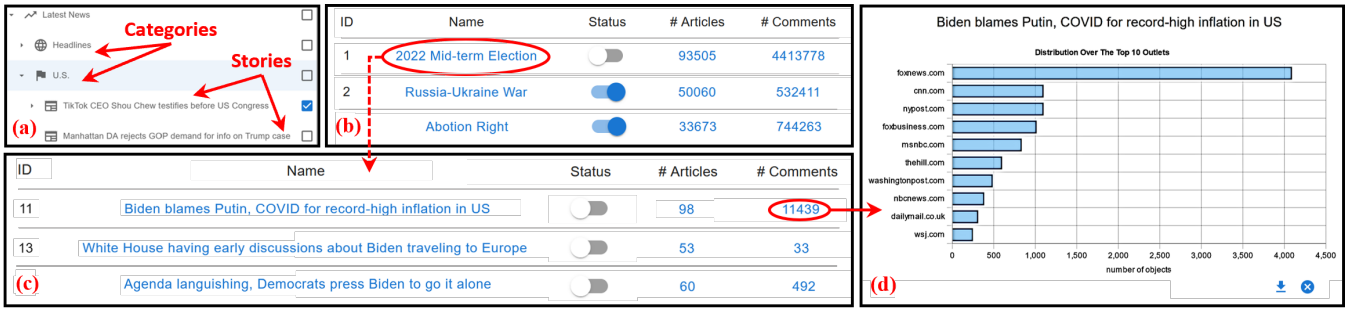

Figure 3: Partial screenshots of the Web portal interface for task management. (a) The news feeds from Google News. Users may select any category/story/article to start a new task. (b) The top-level task table. Each row is a task created by the user. (c) Story tasks under the "2022 Mid-term Election" task. (d) When the article/comment number is clicked, a bar chart will pop up and show the distribution of the objects over the top 10 outlets. The pop-up also provides a download link to the data.

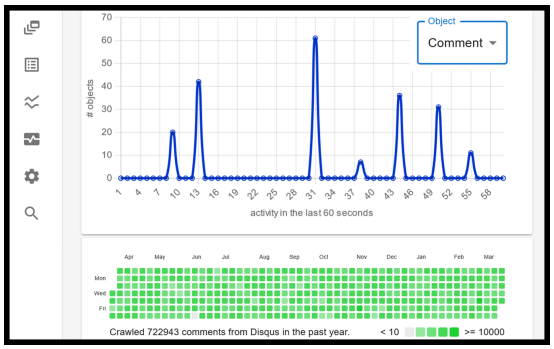

Figure 4: The latest activity (top) and contribution calendar (bottom) of the crawler.

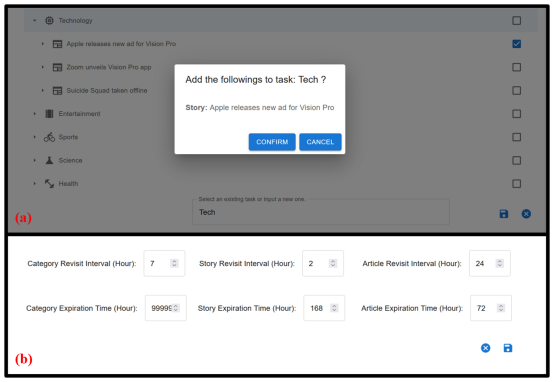

Figure 5: Comquest supports creating tasks on news synced from Google News. The system allows users to (a) add new stories/topics and (b) set crawler parameters on the fly.

News-oriented Crawling. Traditional comment crawlers are generally developed for specific outlets, and it is left for the users to sort out interested content, which can be a heavy burden. As shown by Figure 5 (a), we will show that our Web portal syncs with the latest Google News and it allows users to set up crawling tasks for their interested topics/stories.

Tasks Management. The users can check the status of each task and the harvested comments, as demonstrated in Figure 3 (b), and they can monitor the real-time activity and the crawling history using the crawler monitor (Figure 4). We will show the dynamic nature of our system, which allows users to add/remove new stories/topics to existing tasks (Figure 5 (a)) and change crawling parameters on the fly (Figure 5 (b)).

ACKNOWLEDGMENTS

This work was supported in part by grants from the U.S. NSF 2137846 and 2107213, ARO W911NF-20-1-0254. The views and conclusions contained in this document are those of the authors and not of the sponsors.

REFERENCES

[1] Jumanah Alshehri, Marija Stanojevic, Eduard Dragut, and Zoran Obradovic. 2021. Stay on topic, please: aligning user comments to the content of a news article. In *ECIR*. Springer, 3–17.
[2] Zhijia Chen, Weiyi Meng, and Eduard Dragut. 2022. Web Record Extraction with Invariants. *VLDB* (2022), 959–972.
[3] Alon Halevy and Jane Dwivedi-Yu. 2023. Learnings from Data Integration for Augmented Language Models. *arXiv preprint arXiv:2304.04576* (2023).
[4] Lihong He, Chao Han, Arjun Mukherjee, Zoran Obradovic, and Eduard Dragut. 2020. On the dynamics of user engagement in news comment media. *WIRDMKD* 10, 1 (2020).
[5] Lihong He, Chen Shen, Arjun Mukherjee, Slobodan Vucetic, and Eduard Dragut. 2021. Cannot predict comment volume of a news article before (a few) users read it. In *ICWSM*. 173–184.
[6] Andrey Kolobov, Yuval Peres, Eyal Lubetzky, and Eric Horvitz. 2019. Optimal freshness crawl under politeness constraints. In *SIGIR*. 495–504.
[7] Laks VS Lakshmanan, Michael Simpson, and Saravanan Thirumuruganathan. 2019. Combating fake news: a data management and mining perspective. *VLDB* (2019), 1990–1993.
[8] Qingyuan Liu, Eduard C Dragut, Arjun Mukherjee, and Weiyi Meng. 2015. Florin: a system to support (near) real-time applications on user generated content on daily news. *VLDB* (2015), 1944–1947.
[9] Chen Shen, Chao Han, Lihong He, Arjun Mukherjee, Zoran Obradovic, and Eduard Dragut. 2022. Session-based News Recommendation from Temporal User Commenting Dynamics. In *ASONAM*. IEEE, 163–170.
[10] Luke Sloan, Jeffrey Morgan, William Housley, Matthew Williams, Adam Edwards, Pete Burnap, and Omer Rana. 2013. Knowing the tweeters: Deriving sociologically relevant demographics from Twitter. *Sociological research online* 18, 3 (2013), 74–84.
[11] Ting Wu, Lei Chen, Pan Hui, Chen Jason Zhang, and Weikai Li. 2015. Hear the whole story: Towards the diversity of opinion in crowdsourcing markets. *VLDB* 8, 5 (2015), 485–496.