SM-Scraper: Analysis and Visualization of Scraped Social Media Content

Deepti Bisht, Shuhong Chen, Ophir Gal, Jerry Qian, and Yiheng Xu

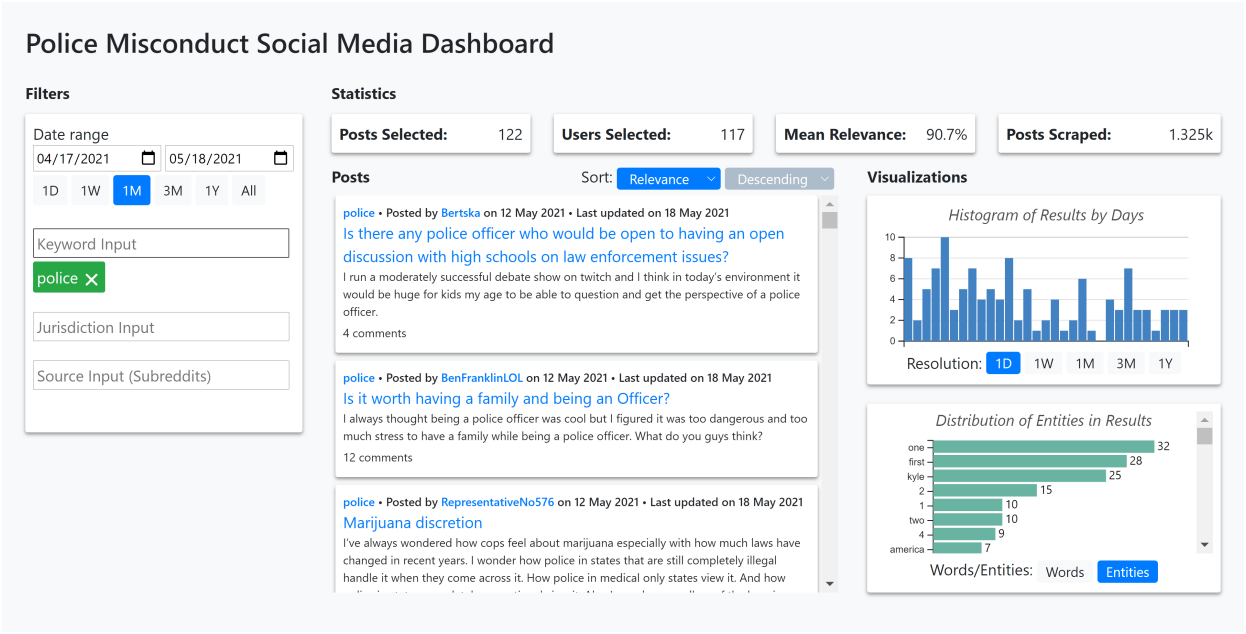

Fig. 1. Screenshot of the Dashboard front-end displaying Reddit posts.

Abstract— With the prevalence of police misconduct and general abuse of power, users of social media platforms have been producing increasing amounts of online content to help raise awareness of such events as well as voice their opinion. The public availability of such data poses an opportunity for meaningful analysis for organizations wanting to make a positive impact. In this paper, we develop a software package, *sm-scraper*, which scrapes text from social media websites, filters and parses it into a database, and visualizes the data through an interactive search engine. The system is designed for the Full Disclosure Project, which aims to identify instances of police misconduct. In consideration of our client, we engineered our application to be easily extendable, and placed emphasis on identifying misconduct-related posts with their corresponding jurisdictions. To complete the project, we overcame various technical challenges associated with the scraper API (specifically, Reddit), lemmatized keyword search, jurisdiction identification from named entities, post relevance classification, and dashboard design.

Index Terms—scraper, visualization, natural language processing, social media, police misconduct, interactive analysis

1 Introduction

With the increase in police brutality and general misconduct, the amount of related social media posts also rises. A police misconduct can be any act which is illegal, unethical or involves violation of individuals' constitutional rights by police officers. Examples of police misconduct include police brutality, dishonesty, sexual assault, abuse of authority and torture or physical abuse in police custody in order to obtain or coerce a confession. Under international law, police officers should only ever use lethal force as a last resort. This means police officers should only apply such force to protect themselves or others from threat of death or serious injury but many killings by the police that we have seen around the world clearly do not meet this criteria. In the USA, George Floyd, Tamir Rice, Daunte Wright, Michael Brown, Breonna Taylor, Eric Garner and too many other African Americans who have been killed by police were unarmed. There has been a lack of transparency and accountability when it comes to police misconduct cases and failure to expose these cases can lead to further increase in police harassment. Our project aims to make police misconduct discussions from social media more accessible and easy to track and analyze. By building a social media scraper, we streamline the process for discovering relevant and up-to-date social media posts in real time. Furthermore, by using an automated scraper, we can gather a much more thorough collection of posts. We hope that with access to this data, data scientists, lawyers, and other relevant parties will gain a better understanding of police brutality cases across the U.S. (and potentially the world depending on scraper settings) and their public perception. In the age of technology and data, people should be able to quickly find examples and discussions of world events. We developed a software package, *sm-scraper*[1], that crawls social media websites for police misconduct posts into a database, for use in an interactive visualization. The project consists of 4 major components: a scraper, a Natural Language Processing (NLP) module, a database, and a dashboard.

[1] *sm-scraper* is available on a public GitHub repository at `https://github.com/Ophir-Gal/sm-scraper`.

Scraper: This component continuously fetches content from social media websites through their respective APIs provided by these websites. We exclusively supported Reddit for this project, but designed and documented the code to be platform-agnostic.

Database: The database is responsible for calling the NLP module to decide whether a post is relevant and then store the relevant posts in the database. We use classifiers based on simple keyword heuristics for our project as an example, but allow users to write custom classifiers.

NLP: This component is responsible for processing scraped text, before writing to the database. In particular, it performs named entity recognition (NER) and lemmatization on the raw text, in preparation forentities, interactive downstream visualization. Spacy is used to perform the NER and lemmatization.

Dashboard: Finally, we conveniently present the contents of the database through an interactive dashboard. The core feature is an advanced search bar where users can specify keywords, jurisdictions, dates, or websites (subreddits for this project). The selected posts then appear as search results with relevant highlights, statistics, visualizations, links, and meta-data.

Our visualization dashboard demonstrates the utility of the data. The scraper gathers new posts as they are uploaded to the internet so that our interface can be used as a live tracking tool for users to find the latest events and discussions.

2 Related Work

Overall, our project aims to use data from social media (we focus on Reddit) to enhance social justice, especially police misconduct. We hope our system works as a better platform to give users cleaner information and easier access to police misconduct information. Research has shown that the social media reflects public opinion and emotion on events, and that it makes an impact on justice. Nicola A. Boothe-Perry [4] in his paper "Friends of Justice: Does Social Media Impact the Public Perception of the Justice System?" stated that social media can have a non-negligible influence on the justice system. Therefore, posts from Reddit can give us better insights in detecting evidence of injustice.

Moreover, Harald Schoen et al. [5] in their paper "The Power of Prediction with Social Media" pointed out that social media offers a significant amount of data with important information hiding behind them. And as stated in the papers, a good data set is important. So after we scrape related Reddit posts, we filter keywords such as "police" and "violence". We also obtain important entities such as locations and timestamps, etc., to create a well-structured dataset. These data is meant to help users of the system make better analyses and predictions.

As far as the actual implementation of our project is concerned, a fundamental component is the scraper module. Scraping websites has been researched in several papers. The vigi4Med Scraper framework [1] gives us a view of a good scraper structure and expected features. We learned how to deal with duplicated data and norms for collecting useful data. Existing works demonstrated scraping data from social media is becoming a more and more popular and effective way for data analytics, however, they also pointed out scraping can be challenging. Oskar and Christoffer [12] indicated that one key reason for the difficulty is that many social media platforms have protections against scraping due to privacy and network traffic concerns. This reminded us to look for platforms and APIs that have better accessibility, and finally helped us to decide to use Reddit and Praw. Other works provide methodology suggestions and critique for social media tools. For example, Batrinca and Treleaven's [3] paper recommended doing data cleansing and applying NLP methods on scraped data. So we incorporated these techniques to make our system more effective and perform more sophisticated content classification.

Aside from scraping data, our aim is to give users a better system to interactively analyze social media content, for which a well-designed dashboard is indispensable. The work of Sarikaya et al. [14] demonstrated a comprehensive guide for designing an effective dashboard. It characterized dashboards by their design goals, levels of interaction, and the practices around them. In our design of the dashboard, we especially referred to it for the layout and styles. Existing works also give guidelines for other detailed implementations. Bertin [9] talked about visual encoding theory, marks, and visual variables, and listed proper encoding channels for different types of data. We referred to it in choosing the right encodings for the plots. Additionally, Jock Mackinlay [13] talked about improving the effectiveness of the visualization by highlighting elements of primary interest using "visual popout". We apply this in our project by highlighting the most important information in the plots and posts related to police brutality. Lastly, as an interactive visualization system, the dashboard should also have good interactions. Heer and Shneiderman [6] introduced the taxonomy of interactions and their usage. According to that, we applied sort, filter, select and other interactions to make our dashboard more interactive.

Our NLP system builds upon classical techniques in computational linguistics. Many of the foundational tasks needed to process the text data comes from implementations readily available from modern libraries; in particular, we use the SpaCy Python SDK [7] to perform tokenization, lemmatization, and named entity recognition. The analysis of post relevance takes inspiration from the information science domain, from which we borrow an implementation of TF-IDF [11]; while TF-IDF is typically used for information retrieval, we adapt it as a feature embedding for our post relevance classifier.

It is important to note that there indeed exist professional open-source platforms to solve our problem. One of the most famous search engine ecosystems is ElasticSearch [2], which uses a Java and NoSQL backend to manage text articles and handle search queries. While these systems are the more scalable option, we unfortunately did not have the time and expertise to use them; future iterations building off this work may benefit from these open-source solutions.

3 Design Goals

In this section we present our main design considerations for the design of *sm-scraper*. We defer discussion of the high-level architecture and technical implementation details to the next two sections.

G1. Modularization and extendability. We wanted the project to be modular and easily extendable, since we were not given very specific requirements by our client, the National Association of Criminal Defense Lawyers (NACDL), and in particular, their Full Disclosure Project[2]. We therefore decided to containerize each software component via the use of Docker.

G2. Clear and intuitive user interface. In terms of the general design of the dashboard component, we preferred an interaction-based dashboard interface over a crowded multi-panel application. We did not know who would be the end user of this project, nor what the exact use-case is, so we wanted a straightforward, approachable platform. When the user visits the dashboard, the most important content should be in the center (in our case, the content of social media posts). Additional information and interaction widgets should be located around it (such as filtering widgets and buttons for altering the plots) to promote clarity and simplicity in the exploration process while also enabling customization and further analysis.

G3. Efficiency and responsiveness. We wanted the platform to be fast and efficient when it comes to data retrieval and interaction. This means designing appropriate database queries and plotting aggregate information, as well as using loading animations to indicate to the user that the system is processing.

G4. Domain-specific dynamic query widgets. Lastly, we aimed to fit the filter widgets to the task at hand. Among other elements, we wanted to let the user search through the database by text features such as keyword and jurisdiction. This entailed designing our database tables in a way which accommodates the features we wanted to include, namely the search inputs.

[2]NACDL is an organization with a mission to act towards a society where all individuals receive fair, rational, and humane treatment within the criminal legal system. Information about their Full Disclosue Project is available at `https://www.nacdl.org/Landing/FullDisclosureProject`

4 ARCHITECTURE

The architecture for *sm-scraper* is depicted in figure 2. We designed the project in a modular fashion, so that the user could decide how to extend each module and keep things self-contained. There are four different modules: the database, the scraper, the dashboard, and the NLP module. Each module has its own dependencies. The scraper relies on the NLP and database modules, and so does the dashboard (relies on NLP and database). The dashboard module is an optional module which serves a website locally for exploring statistics, visualizations, and the content of scraped posts.

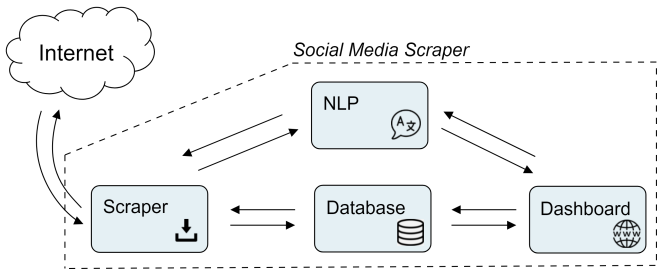

Fig. 2. High-level diagram depicting the different components and relationships in our system.

4.1 Database

We are using psycopg2 to connect to a PostgreSQL database and execute queries. Whenever a post is scraped, the scraper calls the NLP module to get the relevance score of the title and body of the post. If any of them is relevant, the NLP module is called to get the lemmatized version of the post body. A dictionary called "keywords" is created which contains these lemmatized words and their count. Using this "keywords" dictionary , we insert the data to the key_words table. The NLP module is again called to get all entities from the post body and then the values are stored in the "entities" table. In the end, we store the data to scraped_data table. We store "Anonymous" if there is no author for the post and we store the relevance score as the max score out of the relevance score of the title and the relevance score of the body.

4.2 NLP

The NLP module was implemented as a flask server. Calls are made from either the scraper or the dashboard through JSON objects wrapped in HTTP protocol. Users may be interested in writing their own relevance classifiers, which is made possible through the modular design of 'src.nlp.classifiers'; several baseline classifiers are provided for reference, with "all_yes" as the default. We use the SpaCy python library to handle tokenization, lemmatization, and named entity recognition [7], and a SciPy implementation [10] of logistic regression.

4.3 Scraper

The Scraper is the core backend module that scrapes data from Reddit and then stores data into the database. The scraper takes in a list of subreddit names, a frequency to scrape, a limit of the number of posts. It utilizes a Reddit-specific API library called Praw, which allows developers to connect to Reddit and access posts. It provides interfaces for us to get data such as post contents, post author, date published, etc. The Scraper maintains a loop that continuously fetches posts at the frequency to get new posts updated.

4.4 Dashboard

The dashboard is a web application that makes it easy to view and explore the scraped data. The dashboard features a filters panel, a post list, a panel for aggregate statistics, and two visualizations. The user can specify different filters (e.g. a date range) on the left, then view and interact with the results which include the posts in the middle pane, the statistics at the top, and the interactive visualizations on the right.

5 METHODS

In this section we give a more detailed description of each of the four major components in our system. Each component has its own dependencies and was independently developed by different individuals in our team, with the exception of high-level specifications.

5.1 Database

We are using PostgreSQL to store data in 3 tables: scraped_data, key_words and entities. scraped_data is the main table which contains data and metadata of scraped posts that have been classified as relevant to the topic of police misconduct. The NLP module is called to check if a post is relevant according to the given classifier. Thus, only the posts that are relevant are being stored in the database. The schema for the scraped_data table is ["id" Text PRIMARY KEY, "relevance_score" TEXT, "platform" TEXT, "subplatform" TEXT, "time_posted" TEXT, "time_scraped" TEXT, "title_raw" TEXT, "body_raw" TEXT, "author" TEXT, "post_url" TEXT, "comment_count" Int]. The key_words table stores all the keywords present in the relevant posts and their count. The schema for the key_words table is ["id" Text, "key_word" Text, "count" Integer, PRIMARY KEY (id, key_word)]. entities table stores all the entities identified in the relevant post , its type and count. NLP module is being called to get all the entities in a post. The schema for the entities table is ["id" Text, "entity" Text, "type" Text, "count" Integer, PRIMARY KEY (id, entity, type)].
We also use two python scripts: db_backup.py and db_load.py. db_backup.py is used to create the backup of the data in the database in a TSV file and db_load.py can be used to delete the data from postgres and load the data from a TSV file.

5.2 NLP

There are three major tasks for the natural language processing module: tokenization/lemmatization, named entity recognition, and post relevance classification, each of which is described in a subsection below.

5.2.1 Tokenization & Lemmatization

Tokenization and lemmatization are foundational tasks in natural language processing and computational linguistics. Tokenization involves splitting input text into separate words, usually delimited by a space in English, with some special cases for punctuation; this is relatively straightforward, and we use the default tokenizer provided by SpaCy [7]. Lemmatization involves reducing words into a more canonical form; for example, reversing verb conjugation to infinitive ("think", "thought", "thinking" become "think"), or removing plural forms of nouns. Lemmatization is a key component for our search engine application, since naive character-based filtering would remove posts containing the query keyword in a slightly different form. To further broaden the search, we store and query for keywords in lower case.

5.2.2 Named Entity Recognition

Named entity recognition (NER) is another fundamental task in NLP, involving the identification of noun instances that have been given names. For example, a NER system should be able to identify "New Jersey", "The Beatles", and "Joe Biden" as named entities. In our application to identifying police misconduct, NER is extremely useful for parsing names and locations.

Particularly since jurisdiction is critical to our client's use case, we paid special attention to parse location information as precisely as possible. This was particularly challenging to achieve, since social media data is inherently noisy, and the vast majority of posts do not mention a particular jurisdiction. In cases where a county or city is mentioned, it is difficult to match correctly against a complete list of locations in the US. For instance, does "LA" refer to "Los Angeles" or "Louisiana"; which of the 34 "Springfield"s in the US does this post refer to; how should we infer that "NYPD" implies NYC?

In light of these considerations, we made a best-effort attempt at parsing state information where possible. Specifically, we manually define a mapping from different forms of each state ("NJ", "N.J.", "New Jersey", etc.) to their official two-letter abbreviation. For entities

with label GPE or LOC (provided by the vanilla NER system), we perform this mapping if any are contained in the dictionary, and save the resulting new redundant entity with a STATE label.

5.2.3 Post relevance classification

An interesting problem in this space is the detection of police misconduct-related text. Text classification is a well-studied area of natural language processing, with the most famous application being sentiment analysis. Here, instead of deciding whether a post has positive or negative sentiment, we are interested in whether or not it is related to our topic of interest. Such a classifier would be used by the scraper to determine whether a post is relevant enough to store in the database for further analysis, as well as inform a downstream visualization application of a numeric relevance score by which search results can be sorted or filtered.

In order to achieve extendability for our system, we designed a modular API for the user to write their own classifiers and pick which one to use at runtime. These classifiers are located in 'src.nlp.classifiers', and need only to be registered as a classifier with a python decorator to be ready to use. We provide several baseline models for the end user to build off of, including "all_yes", "all_no", "has_police", and "tfidf". The first two are self-explanatory, and "has_police" detects whether given text contains the word "police" (returning the count as the score).

The "tfidf" option is a logistic regression model trained over the TF-IDF embedding [11] of the post document. We train the embedding using 2,531 positive class samples (posts scraped from r/police, r/policebrutality, r/copwatch, r/2020policebrutality, r/SocialJusticeInAction, and r/Bad_Cop_No_Donut) and 13,112 negative class samples (scraped from r/all); for each class, we hold out an additional 128 samples as a test set. A document is a single post, with all tokens lemmatized, stopwords removed, and lower-cased. With this simple baseline model, we are able to achieve 0.82 F1 score at test time; this stands in contrast to the "has_police" classifier, which achieves only 0.56 F1. Please see Fig 3 for the full results.

	tfidf	has_police
acc	0.843750	0.691406
pre	0.988889	0.980392
rec	0.695312	0.390625
f-1	0.816514	0.558659

Fig. 3. Comparison of relevance classification performance between our TF-IDF logistic regression model and a "police" keyword detector.

Additionally, since logistic regression is a simple linear model, we are able to analyze the weights of the learned model to understand which words are important to the classification. Figure 4 shows the 50 heaviest-weighted words in the vocabulary, as well as the 50 least-weighted ones; in this case, the positive weights correspond to contribution to the positive "police misconduct" class, while negative weights indicate words unrelated to the positive class. Despite being such a simple baseline, we are able to achieve decent performance and gather insights from the model.

5.3 Scraper

To cover the specific details of the scraper, the central parts are the scraping loop and organizing data into the database.

Before the loop starts, the scraper first initiates a *praw.Reddit* object and uses authentications to connect to Reddit. Then in the loop, it goes through all subreddits in the list. For each subreddit name, it uses $Reddit.subreddit(subreddit\_name).hot(limit = l)$ to access posts. This function returns a list of the hottest l posts of this subreddit. It keeps scraping every one minute by default. When it sees the same post, it updates the status of the post instead of duplicating it. Furthermore, to clarify, we only scraped several police-related subreddits in the demo, however, the scraper is able to scrape any subreddits on users' demand.

Then for each post, we further extract detailed information and store them into the database. With Praw, we are able to get information such as post title, contents, author, etc. We then call the NLP module to compute the relevance and extract key_words and entities. After that, we formulate SQL queries to update data in the database. During this process, we ensured the data consistency by atomic commit. Moreover, we ensured the data follows the Toxi solution. Details are discussed in the database section.

word	weight	word	weight
police	15.1241	game	-1.6688
cop	11.1297	cat	-1.3542
officer	8.1231	oc	-1.3298
arrest	5.4036	finally	-1.2240
cops	4.3741	anon	-1.2240
sheriff	4.2517	ve	-1.2084
deputy	3.8718	play	-1.1902
shoot	3.8193	israel	-1.1875
law	3.6474	ape	-1.1821
justice	3.3991	win	-1.0327
nypd	3.3425	draw	-1.0309
force	3.3131	doge	-0.9249
enforcement	3.2554	beautiful	-0.9240
department	3.2039	love	-0.9139
county	2.9931	gay	-0.9051
brutality	2.9518	fan	-0.8938
charge	2.8819	cute	-0.8927
video	2.8498	meme	-0.8814
poll	2.8471	baby	-0.8668
crime	2.8276	gop	-0.8479
policing	2.8090	birthday	-0.8455
myanmar	2.7529	change	-0.8291
taser	2.7194	amc	-0.8207
floyd	2.7059	irl	-0.8122
lapd	2.5643	mob	-0.7777
chase	2.5505	insurrection	-0.7753
murder	2.5189	israeli	-0.7398
protest	2.5154	art	-0.7381
black	2.5132	else	-0.7197
shooting	2.4875	twitter	-0.7008
pd	2.4675	cosplay	-0.6937
kill	2.4423	vaccine	-0.6921
serve	2.4373	hour	-0.6782
portland	2.4367	strong	-0.6762
speed	2.4306	moment	-0.6741
protester	2.3942	gaza	-0.6678
suspect	2.3518	pro	-0.6657
trooper	2.3437	safemoon	-0.6607
military	2.2172	poor	-0.6476
driver	2.1600	moon	-0.6457
pig	2.1599	market	-0.6453
drug	2.0751	cool	-0.6425
assault	2.0267	squeeze	-0.6344
jail	2.0258	stone	-0.6334
swat	2.0198	daily	-0.6298
ticket	2.0069	yeah	-0.6280
footage	1.9960	gme	-0.6244
agency	1.9843	blurse	-0.6239
violence	1.9806	paris	-0.6236
prison	1.9657	til	-0.6219

Fig. 4. The highest (left) and lowest (right) weighted words from the "tfidf" classifier's trained logistic regression weights.

Portability: Although during the discussion at the beginning, the Professor and our team all reached an agreement that we should only focus on Reddit, we still explored the portability of our project to other platforms. For other modules, only minor modifications are needed. The NLP and dashboard modules don't really need to change because they are independent of the platform. The database should also stay pretty much the same as our schema design is already quite comprehensive. The worst case is only to add/delete/change one or two attributes. For the scraper module, we state that it's still relatively easy to scrape data from other social media. As discussed above, we are using Praw to scrape Reddit data. Praw is a Reddit-specific API, yet we found there exist similar APIs for other platforms. For example, Tweepy is a popular scraper API for Twitter posts. Similar to Praw, users just need to create an app, get authentication info, and then use API functions to get posts data. The same for facebook-scraper for Facebook. These APIs provide various functions to get detailed post data such as contents, author, created time, etc. Also, since our scraper is written in python, we claim that it's easy to extend our scraper class with these python libraries. Therefore, our project is portable to other platforms.

5.4 Dashboard

The central feature of the dashboard is the list of social media posts related to the project's topic, police misconduct. Each card container in

the post contains summaries of the original Reddit posts, dates posted, author, subreddit, number of comments, and quick links to the actual Reddit pages. Users can sort the posts by relevance, date, and number of comments.

On the left side, we present a filters module. Users can select date ranges of posts they want the dashboard to display. There are also quick select date ranges for easy selection. Next, there is a keyword input. Users can enter in keywords that they want the posts to contain. Similarly, there is also a jurisdiction input, that filters for posts in specific geographic locations. Lastly, users can also select which "subreddits" they want to display the posts of.

At the top of the page, we display a statistics bar, that includes the number of posts selected, users selected, mean relevance of posts selected, and total posts scraped. This gives users an overview of both the posts that were scraped, and those they have selected to view with the filters panel.

On the right side, there are two visualizations. The first is a histogram of posts binned at user selected resolutions. This histogram helps users understand when the posts that are currently being displayed were originally posted. The vertical axis represents the count of posts in each time interval (e.g. each day or each month), the horizontal axis represents time, i.e. if '1D' is selected, the rightmost bars represent the most recent days within the currently specified date range. The second visualization is a bar chart of the frequency of words and entities in currenlty displayed posts. Users can select to display either all of the words or entities which were detected by a classifier in the NLP module. It features a horizontal bar chart with terms in the currently selected posts in descending order (e.g. in figure 1 the most frequent word in the results is "the", as expected).

The flask server handles querying the database. For each query, the server is provided a dictionary of the filter values described earlier. There is a common function that builds the SQL query string used by all components of the dashboard. We do this to ensure all components are displaying values on the same data.

6 Evaluation

We conducted an online user study to evaluate our system and understand its limitations. In particular, we wanted to assess the usefulness and ease-of-use of our dashboard.

6.1 Evaluation Method

We sent out an online survey to a group of classmates and friends, some with technical background and some with less such background, and a total of 3 participants took the survey. Participants were given a quick explanation about the project and the features of the dashboard and then asked to use the dashboard website, and finally fill out a short questionnaire. We used Google's Chrome Remote Desktop browser extension to let users take control of our computers running the project, and Google Forms to implement the online survey. The questions contained in our questionnaire are listed below:

1. Please rate the overall design of the dashboard.
2. Please rate the overall usefulness of the project to a user who's interested in exploring police misconduct social media posts.
3. Please rate the accuracy/relevance of posts when entering keyword / jurisdiction inputs.
4. Please rate the UI in terms of aesthetics.
5. Please rate the UX in terms of ease of use / clarity.
6. Please rate the choice of visualizations (histogram of results with modifiable resolution and horizontal bar chart for distribution of words/entities).
7. Which feature was the most useful to a user who's interested in exploring police misconduct social media posts (Filters, Aggregate statistics, Content of Posts, or Visualization).
8. Which feature was the least useful to a user who's interested in exploring police misconduct social media posts (Filters, Aggregate statistics, Content of Posts, or Visualization).
9. Do you have any additional feedback or suggestions?

Questions 1-6 were Likert-scale questions where users were asked to choose a value between 1 and 5 with 1 representing "Poor" and 5 representing "Great". Questions 7-8 were multiple-choice and the last one was open-ended.

6.2 Evaluation Results

The quantitative results of our user study are shown in figure 5. Qualitative results are included in the appendix. The participants generally had a positive opinion about the major aspects of our dashboard. As far as what participants thought the most or least useful features are (multiple-choice questions), there was absolutely no agreement or trend among the study participants, so we did not include these results. In terms of qualitative feedback, one of the participants suggested we add a filter for ethnicity/race, and another participant suggested that we add other social media platforms as data sources to the project, e.g. Facebook and Twitter.

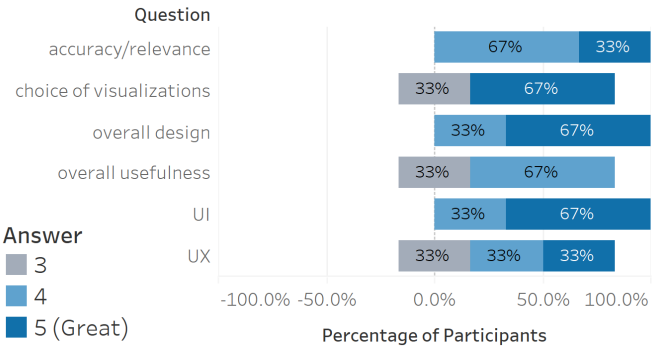

Fig. 5. Likert-scale results from our online user study with questions listed on the vertical axis, percentage of participants on the horizontal axis, and color representing the rating value.

7 Discussion and Future Work

7.1 Scraper & Database

The takeaway from the scraper and database is that the schema should be specifically designed with the end use case in mind. In an earlier version of our schema, we simply stored copies of the lemmatized post titles and bodies along with the raw text. To query all posts for particular keywords, we would have needed to run a regular expression search over space-delimited attribute values. After experimenting (without much success) on using the "LIKE" SQL keyword to perform the filtering, we instead redesigned the schema with a separate Toxi-style tagging system. This not only made the schema cleaner, but also significantly eased the implementation of our dashboard queries.

A key limitation of the scraper is the focus on the Reddit platform. The project requirements left the social media platform vague, and after discussing with Professor Battle, we settled on studying Reddit alone. However, Reddit posts are overall very noisy, and it is difficult to gather key information such as jurisdiction because of the casual nature of the platform. In addition, this means that the information scraped is not held to any standard of accuracy; we simply scrape the "hottest" or most popular posts, which may or may not fail rigorous fact checks. We urge end users to account for the nature of the platform we used, and apply caution before drawing factual conclusions.

One idea for improving the scraper and database backends would be to use open-source publicly-available search engine frameworks. As mentioned in the related work, these frameworks already exist, for example the ElasticSearch API [2]. However, we estimated that it

would have been too difficult for everyone to learn to use the API and use it properly for the project. If we had more time for the project, we would have implemented the system under this framework instead of essentially re-inventing the wheel ourselves.

7.2 NLP Module

The key takeaway from the NLP module is that social media data is extremely noisy. When designing systems that work with text generated by any users around the world, one should take into account that strange text and dubious content will be present to some extent. In our application, it was not uncommon to find special characters (such as emojis) and hyperlinks in the data; special care needed to be taken in order to make sure that this noise did not make it into the analysis.

A major limitation of the NLP module is that we designed it as a separate flask server from the scraper and dashboard. Originally, we had thought that this would have ensured consistency between the NLP API calls for both the scraper and dashboard, and that it would have reduced the burden of having to make the same dependency changes when part of the NLP dockerfile is updated. However, in hindsight, it would have been much more intuitive for the user to be able to import the module regularly as a python module, instead of having to compile a separate docker image and run an independent server at runtime. That said, it does have the advantage of modularization.

Many potential improvements can be made to the relevance classification. In particular, the TF-IDF embedding is a rather naive transformation, and there are more modern embeddings that may or may not improve performance. For example, it has been shown that for sentiment analysis, simply averaging the word embeddings of a BERT transformer can yield excellent results, even surpassing those of syntactic methods [8]. In future iterations, if GPU compute is available to our end user, we may suggest better language models to perform the relevance classification.

7.3 Dashboard

An insight from designing the dashboard is that communication is incredibly important for the final front-end to come together. When first assigning jobs, we divided the project into different components with the intent of working on all of them in parallel and putting the final result together at the end. However, despite our best efforts to design a solid shared API from the onset, it is inevitable that certain design considerations were initially overlooked. Backend schemas and API calls must be very explicitly laid out in a common place for the entire team as early as possible to minimize the amount of confusion and backtracking towards the end of the production cycle.

A limitation of the current system is that it cannot handle datasets of very large scale. As the current implementation stands, all of the post results are listed in a single scrollable list. However, when faced with more than ten thousand posts, it is possible for the host machine to run out of memory. This may be solved in future iterations by paginating the results into a more manageable results box.

A major area of improvement to the dashboard would be guided analysis. Currently, the dashboard offers very few suggestions for the user to begin exploring. The only major component that does this at all is the word/entity distribution histogram, which users may pick interesting keywords from. However, a more targeted design may significantly improve user experience. For example, we could implement auto-complete suggestions in the query boxes, or create a word cloud visualization.

8 Conclusion

Social media is one of the most active places where people discuss police misconduct. To make police misconduct information more accessible, this report presented an interactive analysis system that scrapes data from Reddit and provides a dashboard for efficient analysis. The system consists of four modules: scraper, database, NLP, and dashboard. It scrapes data from any subreddit users are interested in. It uses NLP to figure out the relevance and extract important words. All relevant data is stored in the database in an organized format. The dashboard talks to the database to render plots and posts. It allows users to sort, filter, select the data. Furthermore, we showed that every module can be easily switched to working with other social media platforms. Overall, our four modules harmoniously work together to form a well-designed system that lets users effectively access and analyze police misconduct data, raise awareness, and ultimately improve social justice.

9 Team Member Contributions

Team Member Contributions (1 paragraph) Please describe what each team member contributed to the completion of the project.

- Deepti Bisht:
  - Worked on database part of the project.
  - Listen for data scraped by the scraper and put it into the database.
  - Used NLP flask api written by Shuhong to lemmatize the title and body of the post.
  - Called NLP flask api to calculate the relevant score of the post and stored only the relevant posts in the database.
  - Worked with Yiheng to improve the database design to leverage Toxi solutions for keywords and entities .
  - Worked with Yiheng to write scripts to let user to dump the database content to the tsv file and to upload the data from the tsv file.
  - Worked on the presentation video.
- Shuhong Chen:
  - Set up the project structure with dockerization.
  - Wrote the NLP module, including the flask API, tokenization, lemmatization, named entity recognition, and jurisdiction-specific parsing.
  - Performed word-level analysis of police-related posts.
  - Acted as de facto secretary overseeing meetings and formalizing deliverables.
- Ophir Gal:
  - Designed dashboard layout and features with Jerry.
  - Improved overall dashboard structure layout.
  - Implemented front-end and back-end of “Visualizations” section in dashboard (Histogram, and horizontal bar chart, with interactive widgets).
  - Implemented front-end and back-end of “Statistics” section in dashboard.
  - Worked on overall Docker containerization of the project with Shuhong, in particular, worked to make project cross-platform (Linux, Mac, and Windows).
  - Created survey for evaluation (questions on Google Forms), conducted user study, and visualized the results with Tableau.
  - Created the skeleton for the presentation, created the plots and diagrams for the paper, wrote the evaluation section, design goals section, some of the architecture and methods sections.
- Jerry Qian:
  - Designed dashboard layout and features with Ophir.
  - Implemented overall dashboard structure layout.
  - Implemented Reddit post list module with dynamic loading of posts, sort metrics + ascending/descending.
  - Implemented query filter section:

    - Date range picker with convenient "1D, 1W, 1M, 3M, 1Y, All" button selectors.
    - Keyword, Jurisdiction, and Source pill list input. Interacts with NLP module to lemmatize keywords and convert Jurisdiction states to two letter mode.
    - Implemented SQL query generator that takes filters and converts them into various query modes for consistent posts, stats, and visualizations.
  - Worked on demo portion of presentation video.

- Yiheng Xu:
  - The scraper backend developer.
  - Explored Reddit interfaces and chose Praw to create the client connection to Reddit to scrape data.
  - According to the database schema, figured out the corresponding data in Reddit and the way to access them using Praw.
  - Built the scraper loop that reads in subreddits, keeps scraping data, processes the data, and organizes the data to the database.
  - Wrote scripts to dump/load data from/into the database.
  - Engaged in the design of the database and Toxi solution with Deepti.
  - Explored scraper APIs for other social media such as Twitter and Facebook for platform portability study.

REFERENCES

[1] Bissan Audeh et al. "Vigi4Med Scraper: A Framework for Web Forum Structured Data Extraction and Semantic Representation". In: (2017). DOI: https://doi.org/10.1371/journal.pone.0169658.

[2] Shay Banon. "Elasticsearch". In: (Feb. 2010).

[3] B. Batrinca and P.C Treleaven. "Social media analytics: a survey of techniques, tools and platforms". In: *AI Soc* 30 (2015), pp. 89–116. DOI: https://doi.org/10.1007/s00146-014-0549-4.

[4] Nicola A. Boothe-Perry. "Friends of Justice: Does Social Media Impact the Public Perception of the Justice System?" In: *Symposium: Social Media and Social Justice* 35.1 (2014).

[5] Schoen H. et al. "The power of prediction with social media". In: *Internet Research* 23.5 (2013). DOI: 10.1108/IntR-06-2013-0115.

[6] J. Heer and B. Shneiderman. "Interactive dynamics for visual analysis". In: 10 (Feb. 2012), pp. 30–55. DOI: 10.1145/2133416.2146416.

[7] Matthew Honnibal et al. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303. URL: https://doi.org/10.5281/zenodo.1212303.

[8] Mohit Iyyer et al. "Deep unordered composition rivals syntactic methods for text classification". In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015, pp. 1681–1691.

[9] Bertin Jacques. "Semiology of graphics: diagrams, networks, maps." In: *Cartography* 16.1 (1987), pp. 81–82. DOI: 10.1080/00690805.1987.10438353.

[10] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/.

[11] Karen Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *Journal of documentation* (1972).

[12] Oskar Lloyd and Christoffer Nilsson. "How to Build a Web Scraper for Social Media". In: (2019).

[13] Joke Mackinlay. "Automating the Design of Graphical Presentations of Relational Information." In: *ACM Transactions on Graphics* 5.2 (1986), pp. 110–141. DOI: 0730-0301/86/0400-0110.

[14] A. Sarikaya et al. "What Do We Talk About When We Talk About Dashboards?" In: *EEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 682–692. DOI: 10.1109/TVCG.2018.2864903.

APPENDIX

User Study: Feedback & Suggestions

- Filter by ethnicity/race
- Eventually work with other social media platforms such as twitter and facebook