

**DESAIN DAN IMPLEMENTASI SISTEM CYBER THREAT  
INTELLIGENCE BERBASIS WEB DATA EXTRACTION PADA  
REDDIT**

**Laporan Tugas Akhir**

**Disusun sebagai syarat kelulusan tingkat sarjana**

**Oleh**

**FATHAN ANANTA NUR**

**NIM : 18219008**



**PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
JULI 2023**

**DESAIN DAN IMPLEMENTASI SISTEM CYBER THREAT  
INTELLIGENCE BERBASIS WEB DATA EXTRACTION  
PADA REDDIT**

**Laporan Tugas Akhir**

**Oleh**

**FATHAN ANANTA NUR**

**NIM : 18219008**

**Program Studi Sistem dan Teknologi Informasi**

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Telah disetujui dan disahkan sebagai Laporan Tugas Akhir  
di Bandung, pada tanggal 22 Juni 2023

Pembimbing,

Ir. Budi Rahardjo, M.Sc., Ph.D.

NIP 109110001

## LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Pengerjaan dan penulisan Laporan Tugas Akhir ini dilakukan tanpa menggunakan bantuan yang tidak dibenarkan.
2. Segala bentuk kutipan dan acuan terhadap tulisan orang lain yang digunakan di dalam penyusunan laporan tugas akhir ini telah dituliskan dengan baik dan benar.
3. Laporan Tugas Akhir ini belum pernah diajukan pada program pendidikan di perguruan tinggi mana pun.

Jika terbukti melanggar hal-hal di atas, saya bersedia dikenakan sanksi sesuai dengan Peraturan Akademik dan Kemahasiswaan Institut Teknologi Bandung bagian Penegakan Norma Akademik dan Kemahasiswaan khususnya Pasal 2.1 dan Pasal 2.2.

Bandung, 22 Juni 2023



Fathan Ananta Nur

NIM 18219008

## ABSTRAK

# DESAIN DAN IMPLEMENTASI SISTEM CYBER THREAT INTELLIGENCE BERBASIS WEB DATA EXTRACTION PADA REDDIT

Oleh

FATHAN ANANTA NUR

NIM : 18219008

Ancaman siber telah merambat menjadi masalah sosial. Perusahaan telah banyak berinvestasi dalam pertahanan siber dalam upaya memerangi hal ini. *Cyber Threat Intelligence* (CTI) adalah salah satu mekanisme pertahanan yang akan dibahas. Organisasi baru-baru ini mulai melakukan investasi yang signifikan dalam pengembangan CTI untuk memerangi meningkatnya ancaman serangan siber. CTI dicirikan sebagai sekelompok data faktual yang mencakup konteks, metode ancaman, indikator, dan penanggulangan potensial. Komunitas dan forum *hacker* atau *bad actor* dapat memberikan CTI proaktif yang substansial. Forum, jika dibandingkan dengan platform lain, menawarkan metadata terlengkap, data permanen, dan puluhan *tools*, *technique*, dan *procedure* (TTP) yang dapat diakses secara terbuka. Salah satu platform yang mewadahi komunitas dan forum adalah Reddit. Perlu dibuatkan model atau *prototype* yang menyimulasikan bagaimana informasi dari Reddit diambil. Reddit menyediakan izin untuk mengambil data-data esensial yang dapat dikonversikan menjadi CTI. Praktisi profesional dapat lebih mudah memutuskan tindakan masa depan mereka dengan menggunakan kesimpulan analisis CTI sebagai bantuan visual. Oleh karena itu, data-data yang dikumpulkan dari Reddit dapat diolah dan diberikan visualisasi sesuai konteks, sehingga dapat membantu perusahaan maupun organisasi menyiapkan langkah preventif terkait ancaman siber.

Kata kunci: CTI, Reddit, *web scraping*, *data visualization*

## KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa, sehingga penulis dapat menyelesaikan laporan tugas akhir ini. Laporan dengan judul “Desain dan Implementasi Sistem *Cyber Threat Intelligence* Berbasis *Web Data Extraction* pada Reddit” disusun guna memenuhi persyaratan menyelesaikan mata kuliah Tugas Akhir dan persyaratan kelulusan di Program Studi Sistem dan Teknologi Informasi, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung. Dalam penyusunan laporan tugas akhir, tentunya penulis mendapatkan pengetahuan dan pengalaman dari beberapa pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. I Gusti Bagus Baskara Nugraha, S.T., M.T., Ph.D., selaku Kepala Program Studi Sistem dan Teknologi Informasi,
2. Dr. Fetty Fitriyanti Lubis, S.T., M.T., selaku koordinator Mata Kuliah Tugas Akhir Program Studi Sistem dan Teknologi Informasi,
3. Ir. Budi Rahardjo, M.Sc., Ph.D., selaku dosen pembimbing tugas akhir yang selalu memberikan arahan dan inspirasi,
4. Muhammad Aris Kusnanta dan Nur Hayati selaku orang tua penulis yang selalu memberikan dukungan paling utama dalam penyelesaian tugas akhir,
5. dan segenap pihak yang terlibat dalam pelaksanaan dan penyusunan laporan tugas akhir, sehingga laporan tugas akhir dapat terselesaikan dengan baik.

Penulis menyadari bahwa laporan tugas akhir ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran selalu penulis harapkan, demi penyusunan laporan yang lebih baik lagi kedepannya. Penulis berharap, semoga laporan tugas akhir ini dapat bermanfaat untuk penulis sendiri, dan para pembaca.

## DAFTAR ISI

<b>BAB I PENDAHULUAN.....</b>	<b>1</b>
I.1    Latar Belakang.....	1
I.2    Rumusan Masalah.....	3
I.3    Tujuan .....	4
I.4    Batasan Masalah .....	4
I.5    Metodologi.....	5
I.6    Sistematika Pembahasan.....	6
<b>BAB II STUDI LITERATUR .....</b>	<b>7</b>
II.1    Cyber Threat Background.....	7
II.2    CTI Sharing.....	9
II.3    Model CTI.....	10
II.3.1    Post-Event CTI Sharing .....	11
II.3.2    Pre-Event CTI Sharing .....	11
II.4    Ekstraksi Informasi .....	12
II.5    Forum Internet .....	14
II.6    Reddit.....	14
II.7    Penelitian Terkait.....	17
II.7.1    Studying Reddit: A Systematic Overview of Disciplines, Approaches, Methods, and Ethics (Proferes dkk., 2021) .....	17
II.7.2    Exploring Emerging Hacker Assets and Key Hackers for Proactive Cyber Threat Intelligence (Samtani dkk., 2017) .....	18
<b>BAB III DESAIN DAN IMPLEMENTASI .....</b>	<b>19</b>

III.1	Analisis Permasalahan.....	19
III.2	Rancangan Solusi Secara Garis Besar .....	23
<b>BAB IV PEMBAHASAN, IMPLEMENTASI, DAN EVALUASI .....</b>		<b>27</b>
IV.1	Ekstraksi Data.....	27
IV.1.1	Struktur Reddit.....	27
IV.1.2	Reddit Developer API.....	33
IV.1.3	Python Reddit API Wrapper (PRAW).....	34
IV.1.4	Struktur Data.....	36
IV.2	<i>Virtual Machine</i> .....	40
IV.2.1	Google Cloud Platform (GCP) .....	40
IV.2.2	NoMachine Remote Desktop.....	42
IV.3	Pengujian Kemampuan <i>Crawling</i> Sistem.....	43
IV.3.1	Pengujian Lama Waktu Proses Berdasarkan <i>Network</i> .....	43
IV.3.2	Pengujian Jumlah Item Berdasarkan Lama Waktu Proses .....	45
IV.3.3	Pengujian Ukuran Data Berdasarkan Jumlah Item yang diperoleh .....	48
IV.3.4	Pengujian Konversi JSON ke CSV dan Database .....	51
IV.4	Persiapan Data .....	52
IV.4.1	Penghapusan Baris Ganda .....	52
IV.4.2	Manajemen Data Kosong.....	53
IV.4.3	Pembersihan Data .....	54
IV.4.4	Manajemen <i>Stopwords</i> dan <i>Irrelevant Items</i> .....	55
IV.5	Visualisasi Data .....	57
IV.5.1	Informasi yang Dapat diambil dari Data Mentah Hasil Ekstraksi .	57

IV.5.2	<i>Dashboard Cyber Threat Intelligence</i> .....	69
<b>BAB V KESIMPULAN DAN SARAN .....</b>		<b>71</b>
V.1	Kesimpulan .....	71
V.2	Saran .....	73



## DAFTAR LAMPIRAN

<b>Lampiran A. Source Code Program.....</b>	<b>78</b>
A.1 subreddit_downloader.py .....	78
A.2 dataset_builder.py .....	93
A.3 dataprocess.ipynb .....	102
A.4 dataviz.ipynb .....	113

## DAFTAR GAMBAR

Gambar I-1 Diagram <i>Design Thinking</i> (Dusch, 2022).....	5
Gambar II-1 Segitiga Konseptual Serangan Siber .....	8
Gambar II-2 <i>Web Scraper Tree Diagram</i> .....	13
Gambar II-3 Diagram Skematik Struktur Konten pada Reddit (Medvedev dkk., 2019) .....	15
Gambar III-1 CTI <i>8-step Model</i> (Amaro dkk., 2022) .....	20
Gambar III-2 Diagram Arsitektur Sistem CTI Berbasis <i>Web Scraping</i> pada Reddit .....	25
Gambar III-3 Diagram <i>Use Case</i> Sistem CTI Berbasis <i>Web Scraping</i> pada Reddit .....	26
Gambar IV-1 Tampilan Halaman Utama Subreddit .....	28
Gambar IV-2 Tampilan Halaman <i>Submission</i> .....	30
Gambar IV-3 Tampilan Bagian Komentar pada <i>Submission</i> .....	32
Gambar IV-4 Aplikasi Reddit Developer .....	34
Gambar IV-5 Instance Cloud yang Digunakan.....	41
Gambar IV-6 Tampilan Utama OS Ubuntu .....	41
Gambar IV-7 Tampilan Python Notebook di Server GCP.....	42
Gambar IV-8 Tampilan Utama Aplikasi NoMachine .....	43
Gambar IV-9 Grafik Plot Total Item Berdasarkan Waktu .....	47
Gambar IV-10 Grafik Plot Ukuran data Berdasarkan Jumlah Unit .....	49
Gambar IV-11 Grafik Plot Ukuran Data Berdasarkan Waktu .....	51
Gambar IV-12 Grafik Bar Kata Paling Banyak Muncul.....	58

Gambar IV-13 Grafik Bar <i>Author</i> Paling Banyak Berkomentar.....	59
Gambar IV-14 Kata Paling Banyak digunakan Oleh <i>Author</i> Tertentu .....	61
Gambar IV-15 Jumlah Komentar Per Bulan.....	62
Gambar IV-16 Grafik Jumlah Kemunculan Kata Tertentu Per Bulan.....	64
Gambar IV-17 Grafik <i>Author</i> yang Membicarakan Kata Tertentu.....	66
Gambar IV-18 Grafik <i>Author</i> yang Membicarakan Topik Tertentu.....	68
Gambar IV-19 <i>Dashboard</i> CTI.....	70

## DAFTAR TABEL

Tabel III-1 Kebutuhan Fungsional Sistem .....	22
Tabel III-2 Kebutuhan Non Fungsional Sistem .....	23
Tabel IV-1 Daftar Data Item <i>Submission</i> Reddit API.....	36
Tabel IV-2 Daftar Data Item Komentar Reddit API.....	38
Tabel IV-3 Data <i>Field Submission</i> yang Sudah diparsing .....	39
Tabel IV-4 Data Field Komentar yang Sudah di- <i>parsing</i> .....	40
Tabel IV-5 Data Hasil Percobaan Pengujian Lama Waktu.....	44
Tabel IV-6 Data Hasil Percobaan Pengujian Jumlah Item.....	45
Tabel IV-7 Data Hasil Percobaan Pengujian Ukuran Data .....	48
Tabel IV-8 Konversi JSON ke CSV dan <i>Database</i> .....	51

# **BAB I**

## **PENDAHULUAN**

Bab Pendahuluan pada laporan ini dijadikan sebagai landasan kerja dan arah kerja tugas akhir yang berfungsi untuk mengantarkan pembaca dalam membaca laporan tugas akhir secara keseluruhan.

### **I.1 Latar Belakang**

Kejahatan siber telah berkembang secara signifikan sejak komputer dikembangkan dan dibangun untuk dapat berkomunikasi satu sama lain. Bertepatan dengan paradigma Revolusi Industri 4.0, semakin banyak perusahaan yang menghubungkan pabrik dan infrastruktur bisnis mereka ke internet, yang juga disebut *Industrial Internet*, untuk meningkatkan efektivitas dan efisiensinya. Didorong oleh kekhawatiran yang berkembang akan potensi *vulnerability* pada jaringan dan oleh meningkatnya jumlah gangguan di domain siber, banyak organisasi dan perusahaan mengambil langkah-langkah untuk lebih memahami *vulnerability* dan ancaman yang menjadi sasaran infrastruktur informasi mereka. Terlebih dari itu, banyak organisasi telah memutuskan mengambil langkah-langkah untuk melindungi aset-aset tersebut. Untuk melawan meningkatnya ancaman serangan siber, organisasi dalam beberapa tahun terakhir telah mulai fokus berinvestasi dalam mengembangkan *Cyber Threat Intelligence* (CTI).

CTI pada umumnya merupakan proses yang berbasis data dengan mengumpulkan dan menganalisis data dari sistem internal seperti informasi keamanan dan sistem manajemen *event*, *file log*, sistem deteksi dan pencegahan intrusi jaringan, dan lainnya untuk memberikan wawasan tentang ancaman yang muncul dan aktor ancaman siber. Secara keseluruhan, tujuan CTI adalah untuk meningkatkan kemampuan organisasi dalam menghadapi ancaman siber dan melindungi aset mereka dengan mengumpulkan, menganalisis, dan memanfaatkan

informasi intelijen yang relevan. Terlepas dari nilai dan prevalensinya, data yang dikumpulkan dari sistem internal dianggap CTI reaktif. Untuk mendapatkan data-data sistem internal seperti pada CTI reaktif, diperlukan peristiwa serangan maupun gangguan sebagai objek yang ditinjau. Hal ini tentunya memerlukan pengorbanan tersendiri dari sistem. Untuk mengatasi hal tersebut, perusahaan memerlukan sistem CTI tanpa adanya serangan ataupun gangguan yang masuk ke dalam sistem. Oleh karena itu, istilah CTI proaktif dikenalkan sebagai penyanding CTI reaktif.

Dibandingkan dengan CTI reaktif, CTI proaktif cenderung bersifat preventif dan dilakukan sebelum terjadinya gangguan serangan yang menyebabkan kerugian pada suatu sistem. Informasi mengenai CTI dapat ditemukan dimana saja. Untuk CTI proaktif, informasi secara aktif akan dicari di entitas luar. Terdapat banyak entitas luar yang disinyalir mengandung banyak informasi yang dinilai dapat membahayakan organisasi dan sistem. Media sosial belakangan ini menjadi tren yang digunakan masyarakat umum. Aktor jahat yang secara anonim tergabung ke dalam struktur masyarakat dapat mengakses ke media sosial dan berinteraksi dengan pihak lainnya.

Salah satu penggunaan media sosial yang cukup banyak dilakukan saat ini adalah forum diskusi *online*. Forum diskusi online berbentuk papan pesan online yang dibuat agar para anggota forum dapat berkomunikasi satu sama lain dengan mendiskusikan berbagai topik sehingga memungkinkan untuk saling bertukar pikiran dan pengetahuan (Williams dkk., 2018). Informasi esensial dan sensitif yang menjadi ancaman bagi organisasi terkadang juga dapat dipertukarkan dan mengalir di dalam diskusi online. Sama seperti media sosial lainnya, forum diskusi dapat diakses melalui platform yang bervariasi, salah satu platform yang mudah diakses adalah *website*. Oleh karena itu, sistem *web data scraper* dapat diimplementasikan untuk melintasi struktur tubuh web dan mengumpulkan isinya. Perangkat lunak yang dikenal sebagai *web crawler* menjelajahi internet dengan mengklik tautan dan mengumpulkan halaman web menggunakan protokol HTTP (Fu dkk., 2010).

Salah satu forum diskusi *online* adalah Reddit. Reddit adalah media sosial berjenis agregasi berita, pemeringkatan konten, dan situs web diskusi. Pengguna terdaftar dapat mengirimkan konten ke Reddit berupa tautan, pos teks, gambar, dan video, yang kemudian dilihat dan dipilih oleh anggota lain. Konten diatur berdasarkan subjek ke dalam papan diskusi yang dibuat pengguna yang disebut komunitas atau *subreddits*. Di Indonesia sendiri, Reddit merupakan situs yang diblokir oleh pemerintah dikarenakan konten yang ada di dalamnya dinilai terlalu bebas dan sangat bervariasi termasuk konten pornografi yang dilarang di Indonesia. Dilansir *similarweb.com*, pada November 2022, Reddit menempati posisi ke-20 *website* yang paling sering dikunjungi di seluruh dunia. Oleh karena itu, Reddit dinilai memiliki banyak informasi yang lengkap dan sangat bervariasi baik kontennya maupun penggunanya. Model CTI proaktif cukup cocok jika diimplementasikan ke Reddit karena akan memberikan banyak insight mengenai model dan aktor ancaman ke suatu sistem.

## **I.2 Rumusan Masalah**

Pengembangan sistem intelijen dari data internal adalah fokus utama dari inisiatif CTI saat ini. Hal ini menunjukkan bahwa langkah-langkah keamanan saat ini sering dikelola secara reaktif daripada proaktif. Selain itu, sifat *website* suatu forum membuat metode *web scraper* untuk mengambil data kurang efektif untuk navigasi dan pengindeksan yang efisien. Sehingga, upaya yang dilakukan untuk mendapatkan data dari *website* forum diskusi, sebagian besar terkonsentrasi pada pengumpulan *batch* dan pemrosesan data forum peretas. Koleksi statis ini menjadi kurang bernilai saat ancaman berubah seiring waktu. Maka dengan itu, masalah yang diambil pada artikel ini meliputi

- a. Apakah model *web scraper* dapat diterapkan untuk mengambil dan mengolah data yang dapat dimungkinkan untuk menyusun CTI proaktif?
- b. Apa saja yang bisa diperoleh dari Reddit dengan model *web scraper* yang dibangun?

- c. Bagaimana kinerja model *web scraper* pada Reddit dengan mempertimbangkan masalah yang dihadapi saat pengimplementasian untuk terus mengumpulkan informasi secara kontinu?
- d. Bagaimana visualisasi interaktif dikembangkan dan disajikan kepada akademisi dan praktisi CTI untuk menyelidiki eksploitasi yang dikumpulkan untuk CTI proaktif?

### **I.3 Tujuan**

Penelitian ini dilakukan untuk membuktikan dan memberikan jawaban dari rumusan masalah yang sudah tertulis sebelumnya. Tujuan dari penulisan artikel ini adalah

- a. Membangun model *web scraper* yang dapat diterapkan untuk mengambil dan mengolah data yang dapat dimungkinkan untuk menyusun CTI proaktif.
- b. Mengetahui yang bisa diperoleh dari Reddit dengan model *web scraper* yang dibangun.
- c. Mengetahui kinerja model *web scraper* pada Reddit dengan mempertimbangkan masalah yang dihadapi saat pengimplementasian untuk terus mengumpulkan informasi secara kontinu.
- d. Membuat visualisasi interaktif disajikan kepada akademisi dan praktisi CTI untuk menyelidiki eksploitasi yang dikumpulkan untuk CTI proaktif.

### **I.4 Batasan Masalah**

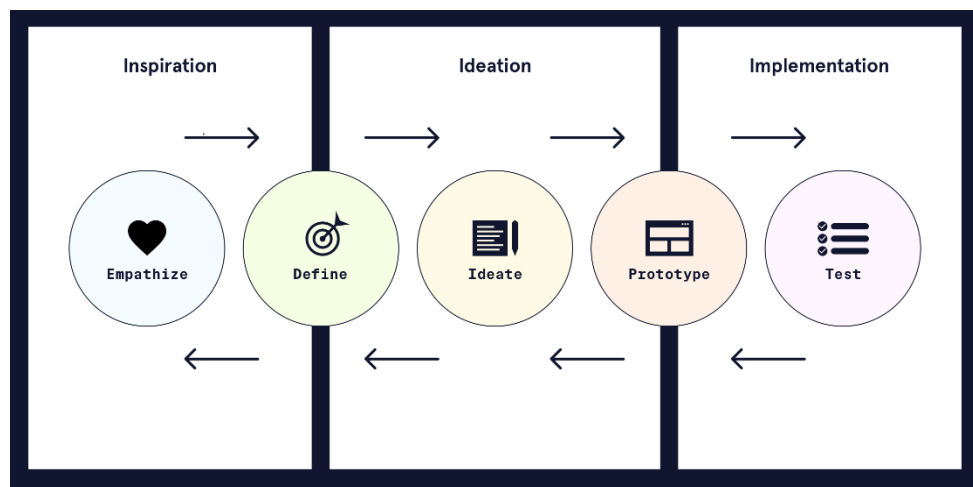
Penelitian ini meliputi pembuatan suatu model *web scraper* yang difungsikan untuk mengambil data yang akan diolah menjadi *insight* CTI yang dapat dianalisis dan digunakan sebagai dasar pengambilan tindakan. Pengimplementasian model *web scraper* dilakukan pada forum diskusi online Reddit. Platform yang dipilih merupakan *website* Reddit sehingga *web scraper* dapat berjalan di atasnya. Penelitian ini menghasilkan suatu *prototype* yang dapat mensimulasikan cara kerja *web scraper* pada Reddit. Fokus topik yang akan dilakukan pengambilan data CTI dari Reddit dilakukan di subreddit tertentu yang memiliki suatu fokus subyek



tertentu pada anggota subreddit yang cenderung homogen. Pembangunan model data scraper pada Reddit mengikuti metodologi dan fase-fase *Software Development Life Cycle* (SDLC) sehingga mudah untuk melakukan evaluasi pada setiap langkahnya. Visualisasi pada penelitian/pekerjaan ini tidak akan diimplementasikan secara kompleks dikarenakan kebutuhan di lapangan dinilai terlalu dinamis dan fleksibel untuk direpresentasikan dalam suatu model.

## I.5 Metodologi

Metode yang digunakan dalam penyusunan sistem ini adalah *design thinking*. *Design thinking* merupakan salah satu metodologi desain yang memberikan pendekatan berdasarkan solusi untuk menyelesaikan masalah. Metode ini dipilih karena pendekatan *design thinking* sangat mengandalkan solusi untuk menjawab suatu permasalahan yang diangkat. Pendekatan semacam ini akan menuntut proses untuk memunculkan sesuatu yang konstruktif demi mengatasi sebuah masalah.



Gambar I-1 Diagram *Design Thinking* (Dusch, 2022)

Design thinking meliputi 5 langkah yang berurutan. Tahap pertama dari proses *design thinking* adalah *empathize* yang memiliki fungsi untuk mendapatkan pemahaman dan empati tentang masalah yang akan diselesaikan. Langkah ini dilakukan untuk mengenali secara menyeluruh tentang permasalahan yang ada. Tahap kedua dalam melakukan *design thinking* adalah *define*. Dalam tahap ini, informasi yang diperoleh dari tahap sebelumnya diolah. Di sinilah informasi

tersebut akan dianalisis dan disintesis untuk mengidentifikasi masalah utama yang akan ditemukan solusinya. Tahap ini menghasilkan pernyataan masalah secara final. Tahap ketiga dari proses *design thinking* adalah *ideate*. Proses *ideate* ini ditandai dengan bermunculannya ide-ide awal dari masalah yang sudah ditentukan sebelumnya. Pada akhir tahap ini, diputuskannya ide terbaik yang akan dibawa untuk memecahkan masalah supaya dapat mempermudah dalam penyusunan pengujian. Tahap selanjutnya adalah *prototype* yang menghasilkan serangkaian versi produk yang diperkecil dan dibatasi atau fitur spesifik yang ditemukan dalam solusi. Solusi diimplementasikan dalam *prototype* akan diteliti dan diperiksa sehingga menghasilkan suatu keputusan bahwa solusi tersebut akan diterima untuk masuk tahap selanjutnya ataupun direvisi dan diperiksa kembali. Tahap *testing* adalah tahap akhir dari *design thinking*, tetapi memungkinkan untuk keseluruhan proses *design thinking* dapat berulang, hasil yang dihasilkan dalam tahap pengujian sering digunakan untuk mendefinisikan kembali satu atau lebih masalah yang ditemukan di tengah proses.

## **I.6 Sistematika Pembahasan**

Laporan ini terdiri dari lima bab. Bab pertama adalah pendahuluan yang berisi subbab sebagai berikut: latar belakang, rumusan masalah tujuan, batasan masalah, metodologi, dan sistematika pembahasan. Bab kedua merupakan studi literatur yang secara garis besar berisi teori-teori terkait dan penelitian terkait. Bab ketiga merupakan desain dan implementasi CTI berbasis *web scraper* pada Reddit. Bab ketiga, secara garis besar, berisi analisis masalah, desain solusi, dan timeline penyelesaian tugas akhir. Bab keempat berisikan evaluasi dari solusi yang diajukan di bab ketiga. Secara garis besar, bab empat membahas tinjauan umum mengenai lingkungan pengerjaan, pengujian beberapa variabel, dan pembahasan hasil pengerjaan. Bab kelima merupakan kesimpulan dan saran yang menjawab permasalahan utama pada laporan tugas akhir serta saran untuk pekerjaan serupa berikutnya.

## **BAB II**

### **STUDI LITERATUR**

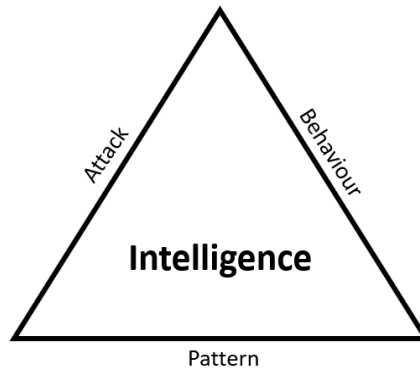
Bab ini mengemukakan landasan metode, teknik, maupun teori yang mendasari, terkait atau yang digunakan pada permasalahan yang dikaji dalam pembangunan Sistem CTI Berbasis *Web Scraper* pada Reddit. Selain itu, bab ini juga menyajikan penelitian serupa yang pernah dilakukan untuk menjaga ketersinambungan ilmu pengetahuan dan *state of the art*.

#### **II.1 Cyber Threat Background**

Seiring berjalannya waktu, entitas organisasi akan selalu menerima tantangan yang signifikan, sehingga diharuskan bagi mereka untuk mempertahankan data dan sistem mereka. Tantangan tercipta dari peningkatan frekuensi dan kecanggihan serangan siber yang diciptakan oleh *threat actor* yang cakap. *Threat actor* yang biasanya bersifat ambisius dan gesit dapat menggunakan berbagai taktik, teknik, dan prosedur (TTP) untuk membahayakan sistem, mengganggu layanan, melakukan tipuan, dan mengungkapkan atau mencuri kekayaan intelektual dan informasi sensitif lainnya (Johnson dkk., 2016). *Threat actor* dapat terdiri dari lingkup individu hingga kelompok sumber daya, bertindak secara sistematis sebagai bagian dari entitas kriminal sampai atas nama negara.

Serangan siber yang dibawa oleh aktor negara dapat menyebabkan kasus politik, diplomatik antar negara yang terlibat dan bahkan menyebabkan kerugian yang lebih besar. Sebagai contoh kasus, pada tanggal 23 Desember 2015, Ukrainian Kyivoblenergo, sebuah perusahaan distribusi listrik regional di Ukraina, melaporkan pemadaman layanan listrik kepada pelanggan. Pemadaman ini disebabkan oleh pihak ketiga yang masuk secara ilegal ke komputer perusahaan dan sistem mereka. Tak lama setelah serangan itu, pejabat pemerintah Ukraina mengklaim pemadaman itu disebabkan oleh serangan siber, dan bahwa dinas keamanan Rusia bertanggung jawab atas insiden tersebut (Lee dkk., 2016).

Berikut ini adalah segitiga konseptual pengenalan serangan siber. Ada tiga komponen utama: *attack*, *behaviour*, dan *pattern* (Almohannadi dkk., 2018).



Gambar II-1 Segitiga Konseptual Serangan Siber

Ide utama segitiga pada Gambar II-1 adalah bahwa kumpulan data insiden siber berisi data serangan, yang dapat dianalisis menggunakan analisis data. Data serangan dapat dipisahkan dari insiden normal dan disajikan dalam format yang lebih mudah dibaca. Jadi, dalam konteks ini, serangan yang dilakukan oleh penyerang mengungkapkan perilaku penyerang. Dengan menambahkan aspek intelijen seperti analisis data, pada dua komponen ini kita dapat mengidentifikasi pola serangan. Pola serangan bisa menjadi kunci untuk mencegah serangan siber di masa depan.

Umumnya, pengumpulan data log sistem dilakukan untuk sebagian besar sistem. Data tersebut dapat dikatakan sebagai *big data* karena data tersebut memenuhi konsep *velocity*, *veracity* dan *volume* (Hilbert, 2015). Jika yang dipertimbangkan hanya volume kumpulan data, maka akan membutuhkan teknik khusus untuk menganalisis dan menyajikannya. Dengan menganalisis data ini, dapat diidentifikasi peristiwa serangan. Peristiwa serangan ini dapat terjadi berulang kali dari waktu ke waktu, yang dapat membentuk pola. Tujuan menggunakan analisis data adalah untuk mengidentifikasi pola seperti itu. Data dapat dianalisis lebih cerdas dan efisien dengan menggunakan teknik analisis *big data*.

## II.2 CTI Sharing

Adanya perkembangan digital yang cepat, bidang serangan yang meluas, dan semakin banyak kerentanan dan metode serangan, entitas bisnis memerlukan lebih banyak langkah dan usaha yang dapat melindungi diri mereka sendiri dan data sensitif mereka. Banyak dari serangan siber yang paling sukses baru-baru ini menimbulkan ide tentang saling berbagi *threat intelligence* untuk menggagalkan dan mitigasi dari serangan. Pemerintah dan pelaku bisnis ingin memberikan cara proaktif untuk mempertahankan diri dari serangan siber dengan membagikan informasi ancaman secara tepat waktu (Feng, 2021). *Intelligence* mengacu pada proses mengumpulkan, menganalisis, dan menafsirkan informasi taktis untuk membuat keputusan. Oleh karena itu, untuk dapat didefinisikan sebagai *intelligence*, informasi harus digabungkan, dianalisis, ditafsirkan, dan disebarluaskan. Informasi mentah, di sisi lain, dapat diperoleh dari segala macam sumber dan dapat menyesatkan, tidak akurat, terputus-putus, dan tidak dapat diandalkan.

Untuk membantu mengurangi serangan siber, perusahaan seperti FireEye dan Cyveillance menyediakan laporan *cyber threat intelligence* (CTI) yang dirancang untuk membantu organisasi melindungi dari serangan siber. Untuk membuat laporan mereka, perusahaan-perusahaan ini mengandalkan data yang dikumpulkan dari serangan atau peristiwa aktual melalui mekanisme seperti log jaringan, log antivirus, *honeypots*, *database access events*, upaya login sistem, dan *intrusion defense system/intrusion protection system* (IDS/IPS) *event log* (Shackleford, 2015).

The SANS Institute memberikan definisi untuk *cyber threat intelligence* sebagai *threat intelligence* yang berhubungan dengan komputer, jaringan dan teknologi informasi (Feng, 2021). CTI mengandung berbagai atribut yang dapat menyajikannya sebagai informasi intelijen. Alamat IP atau hash berbahaya sendiri belum dapat dianggap sebagai CTI, tetapi mereka dapat menjadi bagian darinya. Atribut dapat mencakup deskripsi *threat actor*, manuver, motivasi, dan *Indicator of*

*Compromise* (IoC) yang dapat dibagikan kepada para pemangku kepentingan terpercaya (Wagner dkk., 2019). IoC adalah salah satu atribut CTI yang paling mudah ditindaklanjuti dan merupakan fokus dari sebagian besar model CTI (Farnham dan Leune, 2013). IoC CTI yang dapat ditindaklanjuti biasanya akan diterapkan dalam aplikasi seperti *Intrusion Detection Systems* (IDS), pemblokiran situs web, *blackholing*, mengidentifikasi *host* dan *malware* yang disusupi (Ciobanu dkk., 2014). Teknologi *Big Data* digunakan untuk menyimpan atribut CTI dan terdiri atas keterhubungan antara indikator historis dengan indikator yang baru (Chimson dan Ruks, 2015). Atribut CTI lebih berfokus pada TI perusahaan dan cenderung mengabaikan bidang baru seperti *Internet of Things* (IoT), *Industrial Internet of Things* (IIoT) dan area otomotif. Namun demikian, teknologi-teknologi ini, atau lebih sering disebut sebagai *embedded system*, terhubung ke bagian back-end dan dapat mengambil manfaat dan analisis dari atribut CTI yang ditujukan untuk TI perusahaan.

### II.3 Model CTI

Jaringan perusahaan biasanya akan dilengkapi dengan beberapa perangkat keamanan seperti *firewall* tradisional, IDS, IPS, perangkat lunak *anti-malware*, *traffic sniffer*, dll. Penerapan alat-alat ini merupakan investasi dalam keamanan perusahaan, terutama kemampuan untuk melindungi aset perusahaan dan informasi sensitif. Sebagian besar alat ini adalah sistem deteksi berbasis aturan yang mengizinkan atau menolak lalu lintas data sesuai dengan seperangkat aturan yang harus ditentukan sebelumnya. Di sisi lain, keamanan siber adalah sebuah proses dan *life-cycle*, yang membutuhkan improvisasi berkelanjutan. Oleh karena itu, penting untuk berpikir lebih analitis tentang bagaimana menghadapi ancaman siber tingkat lanjut. Aktif mencari ancaman siber adalah pendekatan yang lebih maju dan kompleks daripada sistem deteksi berbasis aturan tradisional. Informasi ancaman siber dapat diakses dari mana saja, termasuk semua rentang waktu di mana ancaman terjadi. Pada model dibawah, metode dibagi berdasarkan *event time of view* (Almohannadi dkk., 2018).

### II.3.1 Post-Event CTI Sharing

*Cyber threat analysis* adalah kunci untuk melakukan threat hunting. *Cyber threat hunting* adalah proses pencarian potensi ancaman siber melalui jaringan dengan menganalisis kumpulan data yang relevan. Analisis data dapat dilakukan dengan otomatis menggunakan alat yang ada atau sebagai alternatif dilakukan secara manual. Dalam sebuah organisasi, kematangan *cyber threat hunting* bergantung pada kemampuan pengumpulan dan analisis data. Data dapat berupa data historis atau *real-time*, tergantung pada sumber yang dipilih untuk mengidentifikasi ancaman siber. Data ancaman dapat dikumpulkan dengan menggunakan *honeypots* dan dianalisis untuk memahami ancaman sebelum terjadi (Song dkk., 2011). Data juga berisi rincian insiden keamanan siber yang telah terjadi. Menganalisis data tersebut memberikan indikasi bahwa sebagian besar insiden keamanan tidak terjadi sebagai *zero-day attack* (Portokalidis dkk., 2006), mereka cukup sering dan dalam banyak kasus memiliki pola. Pengumpulan dan analisis data yang tepat dapat menghasilkan banyak elemen IoC.

Intelijen yang diberikan bersifat reaktif daripada proaktif, karena didasarkan pada data dari serangan yang telah terjadi. Laporan tersebut belum bisa memberikan informasi intelijen yang komprehensif mengenai alat yang telah dikembangkan oleh peretas dan potensi kerusakan yang besar ketika digunakan untuk serangan siber (misalnya, *zero-day attack*). Selain itu, laporan ini sering mengabaikan aktor tertentu yang bertanggung jawab atas eksploitasi semacam itu, sehingga menghasilkan gambaran yang tidak lengkap tentang lingkungan sudut pandang peretas secara keseluruhan.

### II.3.2 Pre-Event CTI Sharing

*Pre-reconnaissance cyber threat intelligence* mengacu pada informasi yang dikumpulkan sebelum pihak jahat berinteraksi dengan sistem komputer yang ditinjau. Banyak individu di balik operasi serangan siber, yang berasal dari luar laboratorium pendidikan maupun fasilitas militer yang dikelola pemerintah,

bergantung pada komunitas dan forum diskusi yang signifikan. Mereka tetap dapat berinteraksi melalui berbagai asosiasi *online*, tetap anonim, dan menjangkau rekan-rekannya yang tersebar secara geografis. Nunes (2016), pada penelitiannya, berhasil mengembangkan dan menerapkan sistem pengumpulan data intelijen terkait aktivitas aktor berbahaya. Sistem mereka berhasil beroperasi dan sedang dalam proses pengimplementasian sistem ini ke mitra komersial. Menurutnya, masih banyak tantangan desain untuk mengembangkan *crawler* terfokus menggunakan *data mining* dan teknik *machine learning*. Basis data yang dibangun tersedia bagi para praktisi keamanan untuk mengidentifikasi ancaman dan kemampuan siber yang muncul.

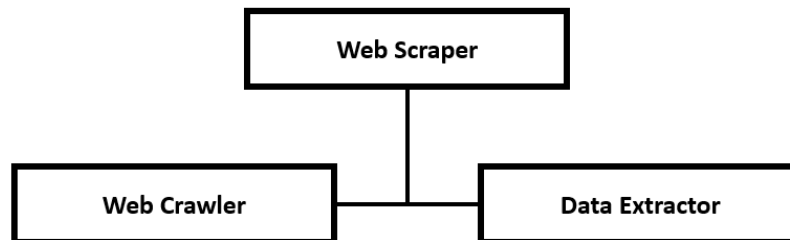
## **II.4 Ekstraksi Informasi**

World Wide Web adalah ruang informasi global yang berisi jutaan data yang dapat diakses melalui Internet, dan mengekstraksi sejumlah besar data dari web dikenal sebagai *web data extraction* atau *web scraper*. Seperti yang dijelaskan oleh Marres dan Weltevrede pada tahun 2013, dalam jurnalisme, *web scraping* juga telah digunakan untuk menilai signifikansi berita internasional dengan menghitung berapa kali berita tersebut disebutkan oleh pengguna media sosial (Marres dan Weltrverde, 2013). Riset di bidang ekstraksi informasi, yang menjelaskan bagaimana konten tidak terstruktur atau semi terstruktur dapat diproses untuk memenuhi tujuan akhir informasi yang ditentukan, telah memungkinkan jenis pemrosesan ini.

Pada tahun-tahun awal, teknik web data scraping yang ada adalah manual *human-copy-paste*. Karena sifat dinamis dan perkembangan teknologi dari dunia web, metode tradisional seperti manual human-copy-paste tidak efektif dilakukan. Oleh karena itu, aspek otomatisasi digunakan dalam proses web scraping. Karena bahasa pemrograman yang digunakan untuk menampilkan halaman web modern, yang dikenal sebagai *Hypertext Markup Language* (HTML), terstruktur secara hierarkis menjelaskan makna teks atau konten lain yang dikandungnya, atau disebut sebagai web semantik, ekstraksi data otomatis dari web dapat dibuat. Web scraper



secara umum terdiri dari dua bagian, yang pertama adalah crawler dan yang kedua adalah ekstraktor data yang ditunjukkan pada Gambar II-2.



Gambar II-2 *Web Scraper Tree Diagram*

*Web crawler*, *crawler* atau *web spider*, adalah program komputer yang digunakan untuk mencari dan secara otomatis mengindeks konten situs web dan informasi lainnya melalui internet. Program ini paling sering digunakan untuk membuat entri untuk indeks pada search engine. Studi membuktikan bahwa algoritma terbaik untuk perayapan web adalah *Genetic Algorithm*. Untuk mengekstrak data besar dan tidak terstruktur dari web ada beberapa teknik dan alat untuk *data extraction* yang dapat dengan mudah mengekstrak dan mengubahnya menjadi format yang bermakna dan terstruktur. Berikut merupakan beberapa cara ekstraksi data (Parvez dkk., 2018).

- a. *Human copy and paste*
- b. *HTML parser*
- c. *HTTP programming*
- d. *Tree-based technique*
- e. *Web Wrapper*

Menurut penelitian yang dilakukan oleh Ferrara pada tahun 2014, secara implisit, *web wrapper* merupakan teknik terbaik untuk mengekstraksi data dari web (Ferrara dkk., 2014).

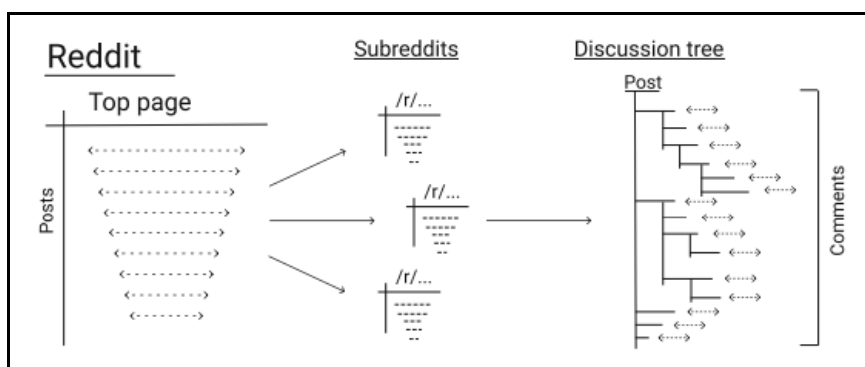
## II.5 Forum Internet

Forum internet adalah papan pesan online yang dibuat agar anggota dapat berkomunikasi satu sama lain dengan mendiskusikan berbagai topik. Meskipun ada banyak kerangka struktur forum, hampir semua forum memiliki struktur tree yang serupa. Sebuah forum dapat mencakup beberapa subforum, yang masing-masing berfokus pada topik yang berbeda (Williams dkk., 2018). Pengguna dapat membuat pos atau diskusi yang relevan dengan subtopik tertentu dalam subforum. Dengan mempos, pengguna terlibat dalam percakapan di sebuah utas. Setiap pos menyertakan nama pengguna *poster*, tanggal pos, dan teks pos. Forum secara otomatis mengarsipkan semua pos; kecuali moderator forum atau *poster* asli menghapusnya, mereka selalu dapat diakses. Forum adalah *platform* berbasis HTML, sama seperti situs web lainnya. Oleh karena itu, *web crawler* dapat digunakan untuk menjelajahnya dan mengumpulkan konten-kontennya. Alat perangkat lunak yang dikenal sebagai web crawler menjelajahi internet dengan mengklik tautan dan mengumpulkan halaman web menggunakan protokol HTTP (Fu dkk., 2010). *Web crawler* tradisional mengadopsi strategi *Breadth-First Search* (BFS) untuk menavigasi *hyperlink*. Akan tetapi, karakteristik dan struktur forum yang unik membuat *web crawler* biasa yang menggunakan strategi ini tidak mempunyai efisiensi yang baik untuk pengumpulan data (Jiang dkk., 2014). Struktur pohon forum sering menghasilkan halaman/tautan duplikat, halaman tidak informatif, dan tautan membalik halaman. Untuk mengatasi masalah ini, studi sebelumnya terutama menggunakan penguraian dan analitik URL yang dikombinasikan dengan ekspresi reguler untuk menargetkan area tertentu dari forum web untuk pengumpulan (Pavkovic dan Protic, 2013).

## II.6 Reddit

Diskusi di forum Reddit pada umumnya bersifat publik karena siapa pun, dengan atau tanpa akun Reddit, dapat melihat konten (kecuali *private subreddit*). Visibilitas konten Reddit yang dibagikan maupun komentar suatu diskusi ditentukan oleh “*voting*” pengguna Reddit. Untuk menjadi pengguna Reddit, yang

dibutuhkan calon pengguna hanyalah memilih nama pengguna yang unik dan kata sandi. Verifikasi email tidak diperlukan untuk membuat akun. Akan tetapi, *terms of service* Reddit mendikte pengguna harus berusia minimal 13 tahun untuk mendaftar. Norma umum di sebagian besar situs cenderung menghindari partisipasi dengan nama asli sebagai tindakan perlindungan privasi. Riwayat partisipasi di situs ini juga bersifat publik, artinya siapapun dapat melihat semua komentar dan kiriman publik pengguna dengan mengklik nama pengguna mereka. Kemudahan yang dapat digunakan pengguna untuk membuat akun memungkinkan, dan tidak jarang, satu orang memiliki banyak akun. Akun “*throwaway*”, atau akun “*dummy*” yang dibuat untuk penggunaan waktu terbatas untuk satu tujuan tertentu, biasanya digunakan saat pengguna tidak ingin konten atau komentar dikaitkan dengan akun utama mereka, seperti berbagi informasi sensitif atau pribadi (Ammari dkk., 2019). Karena partisipasi di Reddit bersifat *pseudonymous*, informasi demografis agak sulit diperoleh. Menurut administrator situs Reddit (Reddit, 2021) mayoritas (58%) pengguna berusia antara 18 dan 34 tahun dan berjenis kelamin laki-laki (57%).



Gambar II-3 Diagram Skematik Struktur Konten pada Reddit (Medvedev dkk., 2019)

Gambar II-3 menunjukkan struktur konten yang ada di dalam Reddit. *User* yang terdaftar dapat mengirimkan *pos* atau *submission* yang berisi judul, tautan eksternal, atau konten yang ditulis sendiri, yang akan langsung tersedia untuk seluruh pengguna Reddit untuk dapat melakukan *voting* dan berkomentar (Medvedev dkk., 2019). Sistem voting hanya mengizinkan pengguna terdaftar

untuk *upvote* (memberikan suara positif +1) atau *downvote* (memberikan suara negatif -1) pada konten atau komentar. Komentar membentuk *discussion tree*, yang dapat digambarkan sebagai akar bercabang, di mana simpul akar utama adalah yang mewakili *post* itu sendiri dan setiap simpul cabang dibawahnya mewakili komentar. Terdapat hubungan antara dua simpul jika ada hubungan “*reply-to*” di antara keduanya. Ruang *pos* utama Reddit dibagi menjadi subreddit yang berisikan komunitas pengguna yang dibuat sendiri dan disatukan oleh topik tertentu. Setiap kiriman yang dikirim memiliki nama subreddit sebagai atribut yang tersirat. Setiap subreddit dan Reddit sendiri memiliki apa yang disebut “*top page*”, sebagai *feed timeline* tempat judul konten dengan tautan *voting* dan komentar dikirim ke *user*. Ada dua faktor yang mempengaruhi posisi peringkat konten, yaitu waktu dan skor *voting*, atau disebut “*karma*”, yang pada dasarnya adalah perbedaan antara *upvote* dan *downvote*. Posting skor tinggi memiliki peluang lebih tinggi untuk muncul di halaman atas.

Subreddit dibuat oleh pengguna dan dimoderasi oleh pengguna. Meskipun ada beberapa aturan Reddit yang menyeluruh tentang konten, subreddit sangat bervariasi mengenai konten yang diizinkan, dan dengan konteks ataupun norma khusus pada bahasan tersebut (Chandrasekharan dkk., 2018). Sebagai bagian dari aturan khusus subreddit mereka, beberapa subreddit memberikan peringatan kepada peneliti tentang pengumpulan data di komunitas. Misalnya, r/depression dan r/SuicideWatch menyatakan semua *pos* dan survei untuk penelitian harus disetujui oleh tim moderator, atau r/IndianCountry yang melarang penelitian tanpa izin dan meminta siapa pun yang tertarik menggunakan subreddit untuk tujuan penelitian harus melengkapi formulir untuk ulasan oleh moderator.

Selain aturan subreddit secara individual, Reddit juga memiliki *user agreement* di seluruh situs. *User agreement* Reddit (Reddit, 2020) mencakup pernyataan sebagai berikut terkait pengumpulan data:

“Access, search, or collect data from the Services by any means [automated or otherwise] except as permitted in

*these Terms or in a separate agreement with Reddit. We conditionally grant permission to crawl the Services in accordance with the parameters set forth in our robots.txt file, but scraping the Services without Reddit's prior consent is prohibited."*

Istilah-istilah ini cukup standar dalam ambiguitasnya (Fiesler, 2020), tetapi pernyataan ini menyiratkan bahwa data yang dikumpulkan di luar batas kelonggaran tertentu dapat dilakukan, misalnya, menggunakan *Application Programming Interface* (API) Reddit. Hal tersebut mungkin merupakan pelanggaran terhadap *user agreement* ini. Namun, API Reddit tersedia secara gratis dan dapat digunakan untuk mengakses konten di situs Reddit.

## **II.7 Penelitian Terkait**

### **II.7.1 Studying Reddit: A Systematic Overview of Disciplines, Approaches, Methods, and Ethics (Proferes dkk., 2021)**

Tujuan dari penelitian ini adalah untuk menjelaskan bagaimana Reddit digunakan oleh para peneliti sebagai sumber data. Pertama, penelitian ini mencatat peningkatan jumlah studi yang menggunakan data Reddit selama sepuluh tahun sebelumnya. Sebagian besar proses dilakukan untuk menghasilkan artikel penelitian yaitu dengan menggunakan teknik komputasi seperti web scraping. Topik yang dipelajari menggunakan data Reddit sangat bervariasi, dan terkadang, peneliti mengambil data dari komunitas di Reddit yang mungkin mencakup populasi yang rentan. Kesimpulannya, hanya sedikit peneliti yang membagikan ilmu yang mereka hasilkan di Reddit, namun hampir 30% penelitian mengenai Reddit di data artikel penelitian ini muncul di Reddit. Ini menunjukkan adanya minat pada Reddit secara luas untuk penelitian tentang Reddit. Namun, eksplorasi lebih lanjut diperlukan untuk lebih memahami nilai yang diciptakan oleh keterlibatan dan berbagi pengetahuan semacam ini.

## **II.7.2 Exploring Emerging Hacker Assets and Key Hackers for Proactive Cyber Threat Intelligence (Samtani dkk., 2017)**

Penelitian ini mengembangkan *framework* baru untuk CTI dengan memanfaatkan pendekatan *mining* baik web, data, maupun teks secara otomatis dan berprinsip untuk mengumpulkan dan menganalisis sejumlah besar *source code hacker*, *tutorial*, dan lampiran langsung dari komunitas internasional hacker bawah tanah yang besar. *Framework* ini memungkinkan peneliti untuk mengidentifikasi banyak aset jahat yang tersedia secara bebas di forum hacker bawah tanah seperti *crypters*, *keyloggers*, *SQL Injections*, dan *cracker password*, beberapa di antaranya mungkin menjadi akar penyebab pelanggaran baru-baru ini terhadap organisasi seperti pada Kantor Manajemen Personalia Amerika Serikat. Peneliti juga dapat menentukan individu utama di balik aset ini dengan menggunakan teknik dan metrik analisis jaringan sosial. Pendekatan dapat digeneralisasikan ke forum peretas mana pun, terlepas dari struktur subforum. Penelitian ini memiliki implikasi praktis bagi organisasi yang ingin meningkatkan *postur* keamanan siber mereka. Dengan asumsi sebuah organisasi mengetahui sistem yang ingin dilindunginya, mereka dapat menerapkan kerangka kerja ini ke forum yang dipilihnya untuk mengidentifikasi aset peretas yang relevan untuk sistem mereka.

## **BAB III**

### **DESAIN DAN IMPLEMENTASI**

Bab ini menjelaskan mengenai analisis permasalahan yang akan dikaji berupa analisis mengenai hambatan-hambatan dari sistem yang sudah dikembangkan. Selain itu, dilakukan juga penjelasan mengenai solusi yang ditawarkan berupa konsep solusi yang akan direalisasikan. Terakhir, akan dijelaskan mengenai timeline penyelesaian tugas akhir.

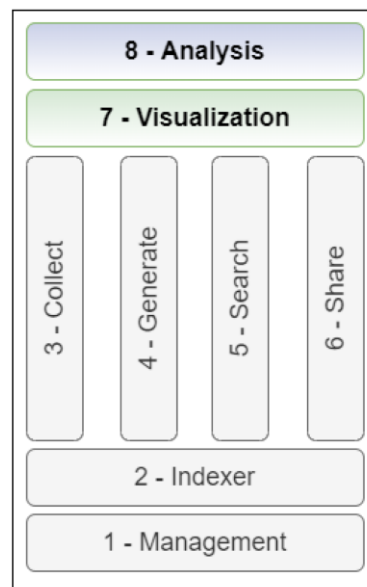
#### **III.1 Analisis Permasalahan**

*Web scraping* sering kali menyerempet *terms of service* yang tertera situs web target. *Terms of service* situs-situs berbasis data yang berat hampir selalu melarang *data scraping* secara ilegal. Melanggar *terms of service* bukan berarti telah melakukan sesuatu yang mengancam diri sendiri. Akan tetapi, pihak situs web tersebut mungkin saja dapat menuntut atas pelanggaran kontrak dan dapat dibawa ke ranah hukum. Terlebih lagi, sebagian besar situs web yang tidak ingin dilakukan *scraping* mempunyai metode penanggulangannya. Web-web itu hanya ingin menyajikan konten kepada pengguna manusia asli yang menggunakan *browser* web, dengan pengecualian jika menyangkut *crawler* milik Google agar terlihat pada pencarian Google. Oleh karena, saat dilakukan *web scraping*, sistem akan menganggap akses dilakukan oleh robot, sehingga terdapat pembatasan. Hal itu terjadi karena sistem tidak menemukan ciri-ciri manusia pada pengaksesannya. Terdapat dua faktor utama untuk dikenali sebagai manusia, yaitu menggunakan peralatan milik manusia (*browser*) atau memiliki perilaku seperti manusia.

Reddit merupakan salah satu platform forum diskusi yang sering digunakan untuk membicarakan berbagai topik. Akan tetapi, terdapat satu permasalahan tentang Reddit yang cukup trivial dan menyangkut hal yang bersifat nonteknis. Reddit merupakan salah satu yang termasuk daftar situs yang dilarang oleh Pemerintah Indonesia. Hal ini mengakibatkan akses ke situs tersebut dibatasi secara

umum. *Proxy* maupun *Virtual Private Network* (VPN) dibutuhkan untuk dapat mengakses Reddit dengan internet publik di wilayah Indonesia.

Terdapat beberapa data points di Reddit yang dapat dilakukan ekstraksi data menggunakan web scraping. *Data points* ataupun sumber data tersebut adalah subreddit, *submission* pada sebuah subreddit, konten yang ada di dalam *post* atau *submission*, seperti: halaman, tautan, komentar, gambar, *upvotes* dan *downvotes*. Selain itu, juga terdapat banyak sumber data lainnya sesuai dengan kebutuhan. Reddit merupakan salah satu *website* yang menyediakan *open source* API untuk beberapa keperluan seperti bisnis dan penelitian. API Reddit dapat dimanfaatkan untuk melakukan *scraping* dengan langsung mengirimkan data dari Reddit ke target tanpa harus mengakses *website* Reddit. Akan tetapi, ada pembatasan bahwa Reddit tidak dapat menyediakan konten untuk dilakukan *scraping* selain 1000 konten teratas atau terpopuler menggunakan Reddit API.



Gambar III-1 CTI 8-step Model (Amaro dkk., 2022)

Kebutuhan pada sistem CTI dapat dijelaskan menggunakan CTI 8-step model yang tertera pada Gambar III-1 (Amaro dkk., 2022). *Step 1 - Management*



bertanggung jawab untuk mengelola interaksi pengguna dengan setiap fungsionalitas yang ditawarkan aplikasi dan interaksi di antara mereka. Step ini memiliki kebutuhan untuk mengontrol aliran data dan aksesnya melalui pengelolaan izin akses. *Step 2 - Indexer* didukung oleh struktur penyimpanan yang mampu mendukung jumlah input dan output data pada sistem CTI. *Step 3 - Collect* bertanggung jawab untuk menyediakan pengumpulan dan penyisipan data eksternal ke dalam sistem CTI. *Step - 4 Generate* dapat menyediakan normalisasi data internal menggunakan pola *Step 2* dan *3* yang sama sehingga data yang diserap dapat diubah menjadi *feeds* untuk digunakan nanti di sistem CTI. *Step 5 - Search* dapat menyediakan mekanisme dan metode untuk memungkinkan manipulasi dan eksplorasi data secara efektif serta memungkinkan pengindeksan data yang cepat dan visibilitas penuh data yang disimpan. *Step 6 - Share* dapat mengizinkan berbagi data internal seperti *feeds*, koleksi, dan indikator ancaman antar pengguna, serta berbagi dengan alat pihak ketiga. *Step 7 - Visualization* dapat memberikan visualisasi data CTI dalam format temporal untuk membuat indikator ancaman sehingga seseorang dapat memiliki gambaran lengkap dari jejak ancaman, IoC, dan informasi berguna apa pun yang diperkaya dan dibagikan oleh pihak lain yang berkepentingan. *Step 8 - Analysis*, yang merupakan bagian intrinsik dari *Step 7*, dapat mengimplementasikan fungsionalitas yang memungkinkan analisis data dan memanipulasi serta memperoleh informasi terbaik dari data ancaman yang tersedia.

Untuk memenuhi *Step 7 - Visualization*, CTI harus dapat menampilkan informasi dan *insight* yang dapat dipahami oleh pihak terkait. Informasi maupun *insight* ini dapat berbentuk sebuah peringatan maupun saran terkait keamanan siber dari sistem yang ditinjau. Akan tetapi, hasil dari *web scraping* masih dalam bentuk *raw data* atau *data frame*. Perlu pengolahan lebih lanjut dari kumpulan data tersebut menjadi informasi yang berharga.

Analisis kebutuhan sistem diperlukan untuk memahami permasalahan dengan melihat gambaran awal dari sistem dan apa saja yang dapat dilakukannya.

Analisis kebutuhan dapat berupa fungsional dan non fungsional. Tabel III-1 dan Tabel III-2 menunjukkan analisis kebutuhan fungsional dan non fungsional.

Tabel III-1 Kebutuhan Fungsional Sistem

ID	Kebutuhan	Deskripsi
FR-1	Sistem dapat mengirimkan API Request ke Reddit	Sistem mengirimkan metode GET ke API Reddit untuk mengambil data terkait konten yang ada di subreddit sesuai dengan tautan yang sudah ditentukan.
FR-2	Sistem dapat menyaring data yang diperlukan	Data yang didapat dari respon API Reddit masih bersifat umum dan berisi semua metadata dan segala isinya. Sistem dapat memilih data berdasarkan parameter yang diinginkan untuk mempermudah pengolahan.
FR-3	Sistem dapat memberikan visualisasi terhadap data yang berhasil didapatkan	Visualisasi dilakukan untuk menyajikan data dalam bentuk yang mudah dipahami sehingga akan mempermudah analisis. Oleh karena itu, sistem dapat menampilkan data yang sudah diubah ke dalam bentuk grafis.
FR-4	Sistem dapat memberikan tanda bahwa subreddit yang dianalisis	Sistem dapat memberikan flag kepada suatu konten hasil dari Reddit <i>Web Scraping</i> . Dengan adanya <i>flag</i> tersebut, CTI dapat memberikan makna kepada pihak terkait untuk memberikan perhatian khusus ke konten tersebut.

	mempunyai level ancaman tertentu	
--	----------------------------------	--

Tabel III-2 Kebutuhan Non Fungsional Sistem

ID	Kebutuhan	Deskripsi
NFR-1	95% <i>response time</i> dari pemanggilan API Reddit tidak lebih dari 500 ms	Rentang waktu yang diberikan ketika mengirim API GET <i>request</i> ke Reddit dan sistem menerima data tidak melebihi 500 ms dalam 95% dari semua kesempatan.
NFR-2	Sistem dapat menyimpan data hasil <i>crawling</i> minimal 1000 data per satu <i>cycle</i>	Data yang disimpan setiap satu <i>cycle web crawling</i> minimal berjumlah 1000 buah. Data yang disimpan disesuaikan dengan aturan Reddit yang hanya mengizinkan maksimal 1000 konten untuk dilakukan <i>scraping</i> .
NFR-3	Seluruh level akses sistem dapat digunakan oleh pengguna	Sistem dapat digunakan sepenuhnya oleh pengguna. Untuk saat ini, level akses belum diberikan spesialisasi karena sistem ini memiliki satu fungsi utama sehingga setiap penggunaan memiliki tujuan yang sama.

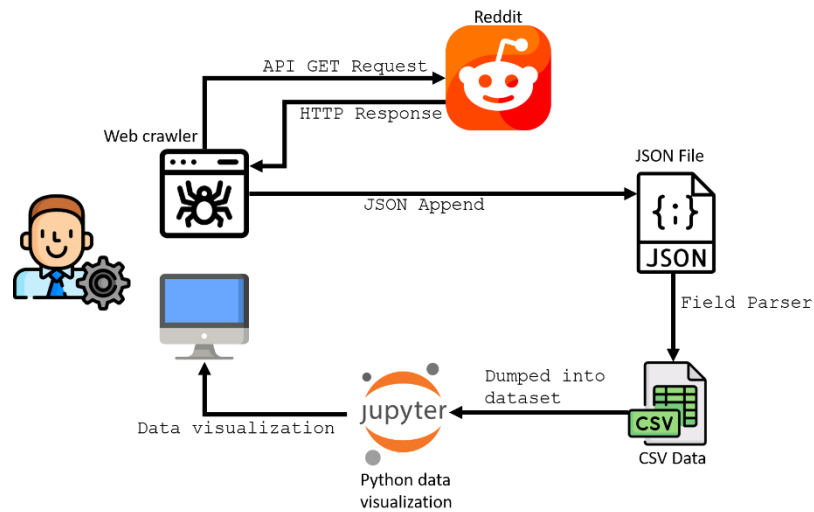
### III.2 Rancangan Solusi Secara Garis Besar

Ada beberapa pendekatan untuk melakukan *web scraping* pada Reddit yang dibagi berdasarkan level keterlibatan pengguna. Pendekatan pertama ialah *scraping*

secara manual. *Scraping Reddit* secara manual adalah cara yang termudah tetapi paling tidak efisien dalam hal kecepatan dan biaya. Akan tetapi, scraping secara manual menghasilkan data dengan konsistensi tinggi. Scraping secara manual cocok untuk kebutuhan scraping yang terbatas hanya untuk beberapa utas Reddit tentang topik tertentu. Pendekatan kedua adalah scraping menggunakan Reddit API. Cara ini dapat menghasilkan data dengan mudah tetapi untuk menjalankannya, diperlukan setidaknya keterampilan dan kompetensi pemrograman. Selain itu, Reddit API membatasi jumlah konten di *data point* mana pun maksimal 1000 buah data. Pendekatan ketiga adalah dengan memanfaatkan layanan API pihak ketiga untuk scraping Reddit. Cara ini adalah pendekatan yang efektif dan mempunyai skalabilitas yang tinggi tetapi menggunakan biaya yang besar. Cara ini hanya dilakukan apabila kebutuhan data dari scraping Reddit melewati beberapa juta konten atau konten.

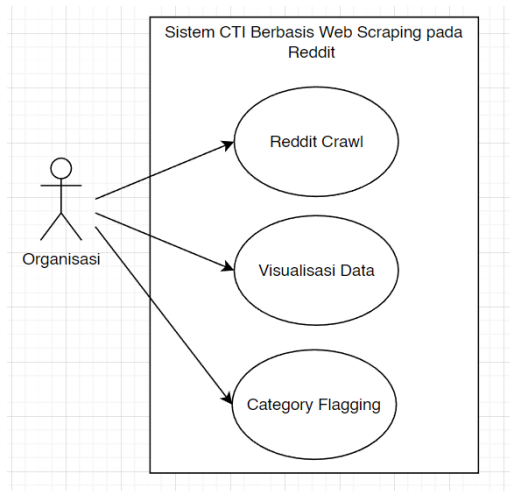
Pembangunan dan implementasi *Reddit Scraper* pada penelitian ini menerapkan pendekatan kedua yakni memanfaatkan API Reddit yang telah disediakan. Selain dapat disesuaikan dengan topik yang akan dilakukan *scraping*, pendekatan ini juga tidak memerlukan server untuk mesin *crawler* ditempatkan. Mesin *crawler* dapat dijalankan di komputer lokal sehingga dapat menghindari pemakaian API Reddit yang tidak wajar dan menghindari penyalahgunaan *terms of service* pada Reddit sendiri.

Untuk menggambarkan keseluruhan sistem *Web Scraping* pada Reddit, dapat digambarkan menggunakan diagram arsitektur.



Gambar III-2 Diagram Arsitektur Sistem CTI Berbasis *Web Scraping* pada Reddit

Gambar III-2 menunjukkan arsitektur sistem CTI berbasis web scraper pada Reddit. Sistem bekerja dengan *web crawler* yang bekerja aktif memanggil *API endpoint* Reddit. *Web crawler* dibangun menggunakan bahasa python dan mengimplementasikan *library* khusus bernama PRAW (Python Reddit API Wrapper). Library ini memungkinkan program untuk berinteraksi dengan Reddit melalui Python. Program/aplikasi *web crawler* akan mengirimkan sebuah *request* GET ke Reddit, lalu Reddit akan merespon dengan mengirimkan konten sesuai dengan tautan subreddit yang tertulis di program. Konten yang dikirimkan dari Reddit masih terdiri dari keseluruhan metadata setiap jenisnya. Untuk visualisasi CTI, hanya diperlukan data-data yang dianggap esensial seperti isi komentar, waktu komentar tersebut dituliskan, atau nama *user*. Oleh karena itu, dari keseluruhan metadata, hanya diambil beberapa untuk dilakukan dumping untuk mendapatkan *dataset*. *Dataset* tersebut dapat divisualisasikan menggunakan *python data visualization* dan disajikan ke pihak yang berkepentingan.



Gambar III-3 Diagram *Use Case* Sistem CTI Berbasis *Web Scraping* pada Reddit

Gambar III-3 merupakan *use case diagram* dari sistem. Sistem mempunyai tiga fungsi utama yaitu Reddit Crawl untuk mengambil data langsung dari Reddit dan Visualisasi Data untuk membuat data dari Reddit dapat dipahami dengan mudah. Pengkategorian tipe ancaman dilakukan juga ketika Visualisasi Data. Data-data yang sudah dikelompokkan saat visualisasi akan diberikan *flag* untuk kata kunci yang sering muncul. Kata-kata yang dipilih sebagai kata kunci adalah kata-kata yang berpotensi mengancam keamanan seperti “*data breach*”, “*password leak*”, “*harass*” dan lain-lain. Jika pada suatu subreddit memiliki banyak *flag* yang sudah dijelaskan di atas, maka akan diberikan nilai dengan skala tertentu berbanding lurus dengan jumlah *flag*. Semakin tinggi skala maka subreddit akan semakin ditandai sebagai ancaman.

## BAB IV

### PEMBAHASAN, IMPLEMENTASI, DAN EVALUASI

Tujuan penulisan bab ini adalah untuk menunjukkan seberapa jauh solusi yang diuraikan pada bagian sebelumnya dapat menyelesaikan permasalahan utama Tugas Akhir. Metode yang dipakai adalah pengujian berdasarkan skenario yang dibangun untuk memvalidasikan kebutuhan yang sudah dituliskan di bab sebelumnya.

#### IV.1 Ekstraksi Data

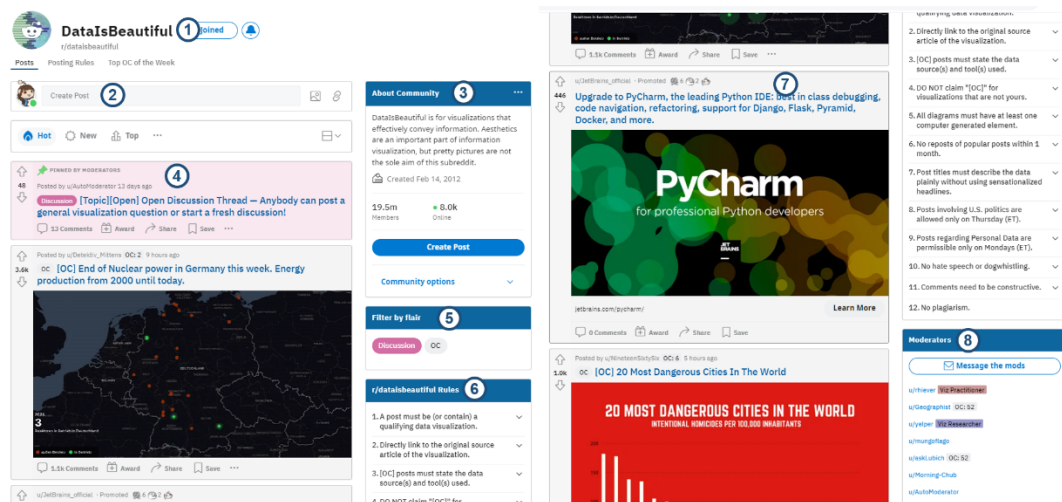
Untuk mendapatkan *dataset* yang nantinya akan dipergunakan untuk analisis dan visualisasi, data diambil secara terpisah dari setiap komentar dan *submission* di dalam Reddit. Data yang diambil secara terpisah tersebut, selanjutnya akan dikumpulkan dan digabungkan dalam satu file tertentu sehingga dapat divisualisasikan dan dianalisis.

##### IV.1.1 Struktur Reddit

Reddit merupakan platform papan pesan *online* yang menyediakan API secara terbuka untuk penggunaanya. Reddit memiliki struktur utama yang terdiri dari *subreddit*, *submission*, dan komentar. Subreddit merupakan forum diskusi yang mengelompokkan berbagai topik dan dapat dibuat oleh pengguna dengan tujuan tertentu. Sedangkan *submission* dan komentar adalah elemen-elemen yang ada di dalam subreddit tersebut. *submission* dapat berupa tautan, gambar, atau teks yang dibagikan oleh pengguna dengan tujuan untuk memulai sebuah diskusi atau memberikan informasi. Sedangkan komentar merupakan tanggapan yang diberikan oleh pengguna terhadap *post* atau komentar yang telah ada sebelumnya. Melalui struktur yang dimilikinya, Reddit memungkinkan para penggunaanya untuk berinteraksi dan berbagi informasi mengenai topik-topik tertentu secara terbuka dan mudah diakses.

#### IV.1.1.1 Subreddit

Subreddit merupakan forum diskusi pada Reddit yang mengelompokkan berbagai topik dengan tujuan tertentu. Subreddit dapat dibuat oleh pengguna dan setiap subreddit memiliki moderator yang bertanggung jawab untuk menjaga agar konten yang diunggah sesuai dengan topik dan aturan subreddit tersebut. Setiap subreddit memiliki nama yang unik dan diawali dengan “r/” diikuti dengan nama subreddit tersebut, seperti contohnya “r/worldnews” untuk subreddit yang membahas berita internasional. Pengguna dapat berlangganan pada subreddit yang diinginkan dan memilih untuk menerima notifikasi ketika ada konten baru yang diunggah di subreddit tersebut. Melalui struktur subreddit yang ada, pengguna dapat dengan mudah mencari dan berpartisipasi dalam diskusi mengenai topik-topik yang sesuai dengan minat dan kebutuhan mereka. Gambar IV-1 di bawah merupakan struktur tampilan subreddit pada versi *website* dan penjelasan setiap bagian.



Gambar IV-1 Tampilan Halaman Utama Subreddit

1. Judul Subreddit: Judul subreddit merupakan nama yang diberikan untuk menggambarkan topik atau tema yang dibahas pada subreddit tersebut. Nama subreddit biasanya mengandung kata-kata yang relevan dengan topik yang dibahas, sehingga memudahkan pengguna untuk mencari dan bergabung dengan subreddit yang sesuai dengan minat mereka.



2. Kolom “*Create Post*”: “*Create post*” merupakan fitur yang disediakan oleh Reddit untuk memungkinkan pengguna untuk membuat konten baru dan membagikannya di subreddit yang relevan (*submission*). Dalam subreddit, pengguna dapat membuat berbagai jenis konten seperti teks, gambar, video, dan tautan ke laman web lain.
3. Tentang subreddit: Halaman di subreddit yang berisi informasi dasar tentang komunitas yang terbentuk di subreddit tersebut. Informasi yang biasanya ada di halaman ini meliputi deskripsi umum tentang subreddit, seperti tanggal pembuatan subreddit dan jumlah anggota.
4. *Post/submission* tersematkan: Fitur yang memungkinkan moderator untuk mengatur dan menampilkan konten tertentu di posisi atas subreddit selama periode waktu tertentu. Konten yang dipasang biasanya berisi informasi penting seperti aturan subreddit, pengumuman, atau *thread* yang sedang populer atau relevan.
5. Filter: “*Filter by flair*” pada subreddit adalah fitur yang memungkinkan pengguna untuk memfilter konten berdasarkan label yang diberikan oleh moderator atau pengguna lainnya. Label ini biasanya digunakan untuk menandai jenis konten tertentu seperti humor, berita, diskusi, tutorial, dan lain sebagainya.
6. Aturan subreddit: Aturan yang dibuat oleh moderator subreddit untuk menjaga kualitas dan tata tertib dalam komunitas. Aturan ini biasanya diatur untuk menghindari konten yang tidak pantas, spam, atau konten yang tidak relevan dengan topik subreddit.
7. *Post/submission: Submission* dalam konteks subreddit pada Reddit merujuk pada konten atau kiriman yang dibagikan oleh pengguna di subreddit tertentu. *Submission* dapat berupa teks, tautan, gambar, atau video yang dapat dilihat dan diakses oleh pengguna lain di subreddit yang sama.
8. Moderator: Anggota yang dipilih oleh pembuat subreddit atau moderator lainnya untuk membantu mengelola dan menjaga kualitas konten yang dipos di subreddit tersebut.

#### IV.1.1.2 Submission

Pada Reddit, konten yang dipos oleh pengguna disebut sebagai “*submission*”. *Submission* dapat berupa gambar, video, teks, atau tautan ke luar situs. *Submission* yang dipos oleh pengguna dapat dilihat oleh anggota lain dari subreddit dan mereka dapat memberikan suara atas *submission* tersebut dengan memberikan *upvote* atau *downvote*. Selain itu, moderator juga dapat melakukan tindakan seperti menghapus *submission* yang melanggar aturan subreddit atau mempromosikan *submission* yang baik ke posisi yang lebih menonjol di halaman utama subreddit. Hal ini menjadikan *submission* sebagai bagian penting dari kegiatan di dalam subreddit dan menjadi fokus utama bagi para pengguna untuk berbagi konten mereka. Gambar IV-2 di bawah merupakan struktur tampilan halaman *submission* tertentu pada versi *website* yang memiliki relevansi dalam *dataset* dan penjelasan setiap bagian.



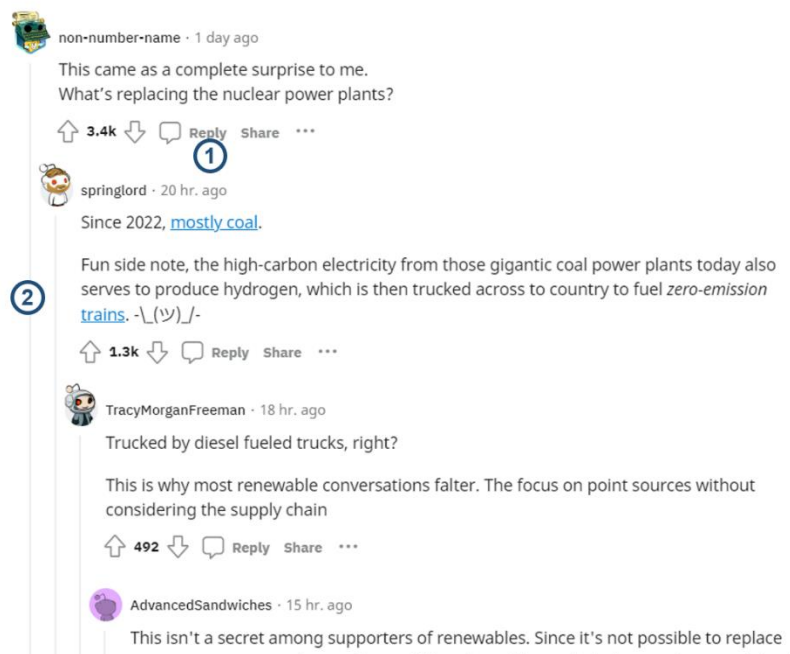
Gambar IV-2 Tampilan Halaman *Submission*

1. *Tanggal post*: *Tanggal post* pada submission Reddit merupakan informasi mengenai tanggal dan waktu pengguna membuat konten atau mengirimkan konten ke subreddit tertentu. Informasi ini umumnya terletak di bagian atas konten atau di samping *username* pengguna yang membuat konten. *Tanggal post* ini sangat penting karena dapat memberikan gambaran tentang seberapa aktif dan terkini sebuah subreddit atau konten tertentu.
2. *Username*: Merujuk pada nama pengguna yang mengirimkan pos tersebut. Setiap pengguna Reddit memiliki *username* unik yang digunakan untuk membedakan satu pengguna dengan pengguna lainnya. Ketika seseorang membuat *submission*, *username* mereka akan tercantum di bawah judul pos, bersama dengan waktu pos.
3. *Upvoted percentage*: Merujuk pada persentase jumlah *upvote* dibandingkan dengan jumlah total vote yang diterima oleh suatu konten. *Upvote* adalah tindakan ketika pengguna menekan tombol panah ke atas untuk menunjukkan bahwa mereka menyukai atau setuju dengan suatu konten. Sedangkan *downvote* adalah tindakan ketika pengguna menekan tombol panah ke bawah untuk menunjukkan bahwa mereka tidak menyukai atau tidak setuju dengan suatu konten. *Upvoted percentage* dapat memberikan gambaran tentang popularitas suatu konten di subreddit dan memberikan indikasi tentang seberapa banyak orang yang menyukai atau tidak menyukai konten tersebut. Semakin tinggi persentase *upvote*, semakin populer suatu konten di subreddit.
4. *Judul*: Teks singkat yang memberikan gambaran tentang isi dari submission tersebut. Judul *submission* biasanya dibatasi oleh jumlah karakter tertentu untuk memastikan agar judul tidak terlalu panjang dan mudah dipahami oleh pengguna. Judul juga bisa digunakan sebagai kunci pencarian yang membantu pengguna dalam menemukan *submission* yang sesuai dengan topik yang dicari.
5. *Komentar*: Tanggapan atau pesan yang ditambahkan oleh pengguna lain pada sebuah *submission*. Pengguna dapat memilih untuk membalas

submission dengan komentar yang bersifat mendukung atau tidak setuju dengan isi *submission* tersebut. Setiap pengguna dapat menambahkan komentar ke suatu *submission*, dan pengguna lain dapat memberikan *upvote* atau *downvote* pada komentar tersebut.

#### IV.1.1.3 Komentar

Komentar adalah salah satu fitur utama pada platform Reddit. Pengguna dapat memberikan tanggapan terhadap sebuah konten yang ada pada subreddit dengan cara menuliskan komentar pada kolom yang tersedia. Komentar dapat digunakan untuk memberikan pendapat, saran, maupun informasi tambahan yang berkaitan dengan topik yang sedang dibahas. Selain itu, komentar juga dapat digunakan untuk memperluas diskusi dan menjawab pertanyaan-pertanyaan yang mungkin muncul dari sebuah konten. Gambar IV-3 di bawah merupakan struktur tampilan halaman komentar pada *submission* tertentu pada versi *website* yang memiliki relevansi dalam *dataset* dan penjelasan setiap bagian.



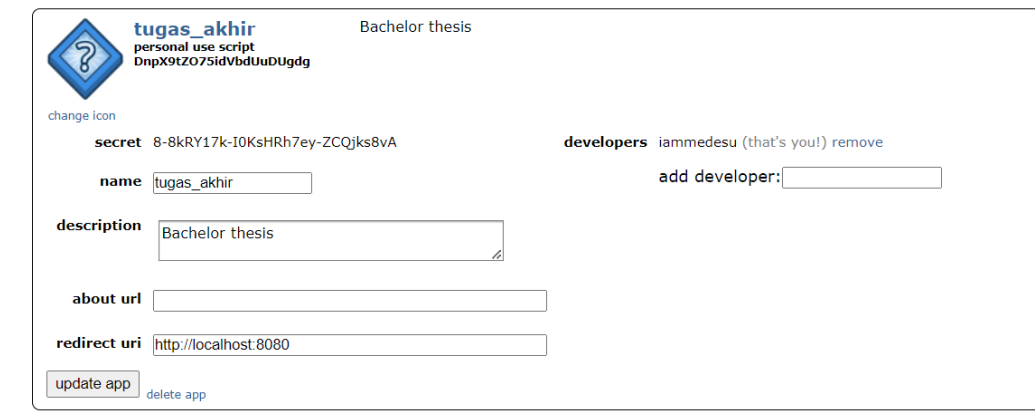
Gambar IV-3 Tampilan Bagian Komentar pada *Submission*

1. *Reply*: Fungsi untuk membalas komentar yang telah dipos oleh pengguna lain. Ketika seorang pengguna ingin memberikan tanggapan atau menjawab pertanyaan dari suatu komentar, mereka dapat menggunakan fitur “Reply” untuk mempos balasan mereka di bawah komentar tersebut. Balasan ini akan terlihat sebagai sub-komentar dan akan memiliki struktur hierarkis di bawah komentar utama.
2. *Reply tree*: Struktur visualisasi hierarki yang menunjukkan hubungan antara komentar dan balasannya dalam sebuah *thread* atau topik diskusi. Setiap balasan yang dipos oleh pengguna akan muncul di bawah komentar induk atau komentar yang ia balas. Jika ada pengguna lain yang menanggapi balasan tersebut, maka komentar baru tersebut akan muncul di bawah balasan yang pertama dibuat.

#### **IV.1.2 Reddit Developer API**

Reddit Developer API adalah sebuah *application programming interface* (API) yang disediakan oleh Reddit bagi para pengembang untuk mengembangkan aplikasi, bot, atau layanan berbasis Reddit. API ini memberikan akses ke berbagai informasi dan data dari platform Reddit, seperti informasi pengguna, pos, komentar, serta informasi tentang subreddit dan moderasinya. Ekstraksi data di Reddit juga menggunakan layanan tersebut. Untuk dapat menggunakan layanan tersebut, pengguna Reddit harus membuat aplikasi di Reddit Developer sehingga Reddit dapat memberikan *personal script* dan *secret code*.

developed applications



The screenshot shows the 'developed applications' section of the Reddit Developer portal. It displays a single application named 'tugas\_akhir' with the description 'Bachelor thesis'. The application is associated with a 'personal use script' and a 'secret' key. The developer listed is 'iammedesu (that's you!)'. There are input fields for 'name' (tugas\_akhir), 'description' (Bachelor thesis), 'about url', and 'redirect uri' (http://localhost:8080). Buttons for 'update app' and 'delete app' are visible at the bottom.

Gambar IV-4 Aplikasi Reddit Developer

Gambar IV-4 merupakan aplikasi untuk ekstraksi data yang didaftarkan ke Reddit Developer. *Personal use script* dan *secret* digunakan untuk mengakses Reddit API yang diimplementasikan menggunakan kode program Python.

### IV.1.3 Python Reddit API Wrapper (PRAW)

Python Reddit API Wrapper (PRAW) merupakan paket *library* bahasa pemrogramman Python yang memungkinkan akses sederhana ke API Reddit.

#### IV.1.3.1 Konfigurasi PRAW

Untuk memulai dan melakukan modifikasi program menggunakan *library* PRAW, dibutuhkan *instance class* Reddit. Terdapat dua jenis instance yaitu: *read-only* dan *authorized*.

##### IV.1.3.1.1 Read-only Instance

Untuk membuat Reddit *read-only instance*, dibutuhkan 3 jenis data, yaitu *client ID*, *client secret*, dan *user agent*. Sehingga inisiasi *class* diekspresikan sebagai berikut.

```
reddit ← praw.Reddit {  
    client_id ← "personal use script",  
    client_secret ← "secret code",  
    user_agent ← "user agent",  
}
```

#### IV.1.3.1.2 Authorized Instance

Untuk membuat Reddit *authorized instance*, diperlukan dua informasi tambahan ke dalam inisiasi *class*. Sehingga inisiasi *class* dapat diekspresikan sebagai berikut.

```
reddit ← praw.Reddit {  
    client_id ← "personal use script",  
    client_secret ← "secret code",  
    password ← "password akun Reddit",  
    user_agent ← "user agent",  
    username ← "username akun Reddit",  
}
```

*Authorized instance* memiliki jangkauan yang lebih luas untuk mengakses informasi Reddit dibandingkan *read-only instance* termasuk menuliskan pos ke Reddit. Namun, sistem yang dibangun saat ini, hanya menggunakan *read-only instances* karena hanya memerlukan pengambilan data yang bersifat *public* dari Reddit.

#### IV.1.3.2 Pengambilan Data Submission dari Subreddit

Setelah membuat *instance* subreddit, iterasi dapat dilakukan melalui *submission* di dalamnya. Setiap *submission* memiliki *instance* tersendiri yang tersusun dari beberapa data. Beberapa jenis *instance submission* mencakup metode sebagai berikut: *controversial*, *gilded*, *hot*, *new*, *rising*, dan *top*. Masing-masing metode ini akan segera melakukan *return* ListingGenerator, yang akan diiterasi di giliran berikutnya. Misalnya, melakukan iterasi melalui 10 kiriman pertama menggunakan *hot sort* untuk subreddit tertentu.

#### IV.1.3.3 Pengambilan Data Komentar dari Submission

Submission memiliki atribut komentar yang merupakan instance CommentForest. Instance itu dapat dilakukan iterasi dan merepresentasikan komentar top-level dari submission dengan jenis comment sort default. Jika ingin mengiterasi semua komentar sebagai list yang diratakan, metode *list()* dapat dipanggil pada instance CommentForest.

#### IV.1.4 Struktur Data

Data yang diperoleh dari API Reddit merupakan *instance* dari sebuah gabungan data yang menyusun sebuah item *submission* maupun komentar. Diperlukan seleksi data-data yang relevan ke dalam satu *dataset* sehingga memudahkan untuk memahami data tersebut.

##### IV.1.4.1 Raw Data

Terdapat data *submission* dan komentar yang diekstraksi menggunakan API Reddit (Reddit.com, 2023). Data tersebut memiliki struktur yang mirip. Hubungan antara data *submission* dan data komentar adalah *one-to-many*, sehingga satu *submission* dapat memiliki beberapa komentar. Hal tersebut dapat dibuktikan bahwa di dalam data komentar terdapat atribut *submission* yang berfungsi sebagai *foreign key*.

##### IV.1.4.1.1 Submission

Tabel IV-1 menunjukkan atribut apa saja yang terdapat dalam satu item *submission* dari Reddit API.

Tabel IV-1 Daftar Data Item *Submission* Reddit API

Attribute	Description
<code>Author</code>	Provides an instance of <code>Redditor</code> .
<code>Author_flair_text</code>	The text content of the <i>Author</i> 's flair, or <code>None</code> if not flaired.
<code>clicked</code>	Whether or not the submission has been clicked by the client.
<code>comments</code>	Provides an instance of <code>CommentForest</code> .
<code>created_utc</code>	Time the submission was created, represented in Unix Time.
<code>distinguished</code>	Whether or not the submission is distinguished.
<code>edited</code>	Whether or not the submission has been edited.
<code>id</code>	ID of the submission.



Attribute	Description
<code>is_original_content</code>	Whether or not the submission has been set as original content.
<code>is_self</code>	Whether or not the submission is a <i>selfpost</i> (text-only).
<code>link_flair_template_id</code>	The link flair's ID.
<code>link_flair_text</code>	The link flair's text content, or <code>None</code> if not flaired.
<code>locked</code>	Whether or not the submission has been locked.
<code>name</code>	Fullname of the submission.
<code>num_comments</code>	The number of comments on the submission.
<code>over_18</code>	Whether or not the submission has been marked as NSFW.
<code>permalink</code>	A permalink for the submission.
<code>poll_data</code>	A <code>PollData</code> object representing the data of this submission, if it is a poll submission.
<code>saved</code>	Whether or not the submission is saved.
<code>score</code>	The number of upvotes for the submission.
<code>selftext</code>	The submissions' selftext - an empty string if a link <i>post</i> .
<code>spoiler</code>	Whether or not the submission has been marked as a spoiler.
<code>stickied</code>	Whether or not the submission is stickied.
<code>subreddit</code>	Provides an instance of <code>Subreddit</code> .
<code>title</code>	The title of the submission.
<code>upvote_ratio</code>	The percentage of upvotes from all votes on the submission.
<code>url</code>	The URL the submission links to, or the permalink if a <i>selfpost</i> .

#### IV.1.4.1.2Komentar

Tabel IV-2 menunjukkan atribut apa saja yang terdapat dalam satu item komentar dari Reddit API.

Tabel IV-2 Daftar Data Item Komentar Reddit API

Attribute	Description
<code>Author</code>	Provides an instance of <code>Redditor</code> .
<code>body</code>	The body of the comment, as Markdown.
<code>body_html</code>	The body of the comment, as HTML.
<code>created_utc</code>	Time the comment was created, represented in Unix Time.
<code>distinguished</code>	Whether or not the comment is distinguished.
<code>edited</code>	Whether or not the comment has been edited.
<code>id</code>	The ID of the comment.
<code>is_submitter</code>	Whether or not the comment <i>Author</i> is also the <i>Author</i> of the submission.
<code>link_id</code>	The submission ID that the comment belongs to.
<code>parent_id</code>	The ID of the parent comment (prefixed with <code>t1_</code> ). If it is a top-level comment, this returns the submission ID instead (prefixed with <code>t3_</code> ).
<code>permalink</code>	A permalink for the comment. <code>Comment</code> objects from the inbox have a <code>context</code> attribute instead.
<code>replies</code>	Provides an instance of <code>CommentForest</code> .
<code>saved</code>	Whether or not the comment is saved.
<code>score</code>	The number of upvotes for the comment.
<code>stickied</code>	Whether or not the comment is stickied.

Attribute	Description
<code>submission</code>	Provides an instance of <code>Submission</code> . The submission that the comment belongs to.
<code>subreddit</code>	Provides an instance of <code>Subreddit</code> . The subreddit that the comment belongs to.
<code>subreddit_id</code>	The subreddit ID that the comment belongs to.

#### IV.1.4.2 Parsing Data

Data mentah yang didapat dari *submission* dan komentar akan dipilih mana saja yang relevan untuk dianalisis. Pemilihan langsung dilakukan dengan program dan otomatis memasukkannya ke dalam file *comma separated value* (CSV).

##### IV.1.4.2.1 Parsing Data Submission

Berikut merupakan data *submission* yang diseleksi dan disimpan ke dalam file CSV.

Tabel IV-3 Data *Field Submission* yang Sudah diparsing

Kolom	Deskripsi
id	Sebagai pembeda antar <i>submission</i> di dalam tabel
created_utc	Waktu <i>submission</i> tersebut dibuat (dalam bentuk <i>Epoch Unix Time</i> )
<i>Author</i>	<i>Username</i> pengguna reddit yang membuat <i>submission</i>
num_comments	Jumlah komentar yang diberikan di <i>submission</i>
title	Judul <i>submission</i>
selftext	<i>Self text submission</i>
full_link	URL <i>submission</i>

#### IV.1.4.2.2 Parsing Data Komentar

Berikut merupakan data komentar yang diseleksi dan disimpan ke dalam file CSV.

Tabel IV-4 Data Field Komentar yang Sudah di-parsing

Kolom	Deskripsi
id	Sebagai pembeda antar komentar di dalam tabel
submission_id	ID sebuah <i>submission</i> yang diberikan komentar
created_utc	Waktu komentar tersebut dibuat (dalam bentuk <i>Epoch Unix Time</i> )
<i>Author</i>	<i>Username</i> pengguna reddit yang memberikan komentar
score	Akumulasi jumlah <i>upvotes</i> dikurangkan dengan <i>downvotes</i>
body	Isi dari komentar tersebut
parent_id	Menandakan <i>branch</i> komentar tersebut
permalink	URL yang langsung merujuk kepada komentar tersebut

### IV.2 Virtual Machine

*Virtual Machine* (VM) digunakan untuk menggantikan sistem lokal dalam menjalankan proses *crawling*/ekstraksi data dari Reddit. Penggunaan VM diharapkan memiliki kinerja yang lebih baik daripada menggunakan sistem lokal dikarenakan memiliki resource yang lebih baik. Selain itu, VM dapat berjalan dalam waktu yang relatif lebih lama, sehingga dapat melakukan ekstraksi data dalam jumlah banyak.

#### IV.2.1 Google Cloud Platform (GCP)

Google Cloud Platform (GCP) adalah layanan yang disediakan untuk mendukung operasional perusahaan IT dan pengembang aplikasi. Google Cloud menawarkan layanan untuk komputasi, penyimpanan, jaringan, *big data*, *machine learning*, dan IoT, serta pengelolaan *cloud*, keamanan, dan *developer tools*. Gambar IV-5

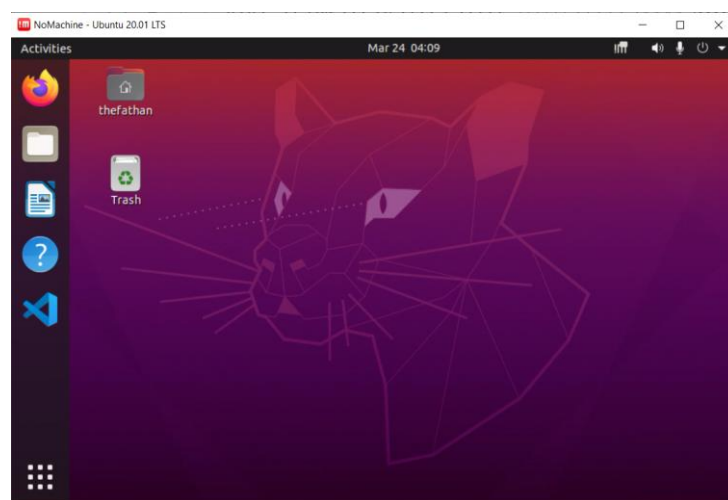
merupakan beberapa instans yang berjalan di Google Cloud Platform untuk pengerjaan Tugas Akhir. Terdapat dua instans yang berjalan, masing-masing adalah Ubuntu Virtual Machine dan Jupyter Notebook Server.

VM instances									
<a href="#">CREATE INSTANCE</a> <a href="#">IMPORT VM</a> <a href="#">REFRESH</a> <a href="#">HELP A</a>									
<a href="#">INSTANCES</a> <a href="#">OBSERVABILITY</a> <a href="#">INSTANCE SCHEDULES</a>									
VM instances									
Filter Enter property name or value									
<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect	
<input type="checkbox"/>	✓	<a href="#">jupyter-ta</a>	asia-southeast2-a			10.184.0.2 ( <a href="#">nic0</a> )	34.101.73.104 ( <a href="#">nic0</a> )	SSH	⋮
<input type="checkbox"/>	✓	<a href="#">ubuntu-scrape</a>	us-west14-b			10.182.0.3 ( <a href="#">nic0</a> )	34.125.183.161 ( <a href="#">nic0</a> )	SSH	⋮

Gambar IV-5 Instance Cloud yang Digunakan

#### IV.2.1.1 Ubuntu Virtual Machine

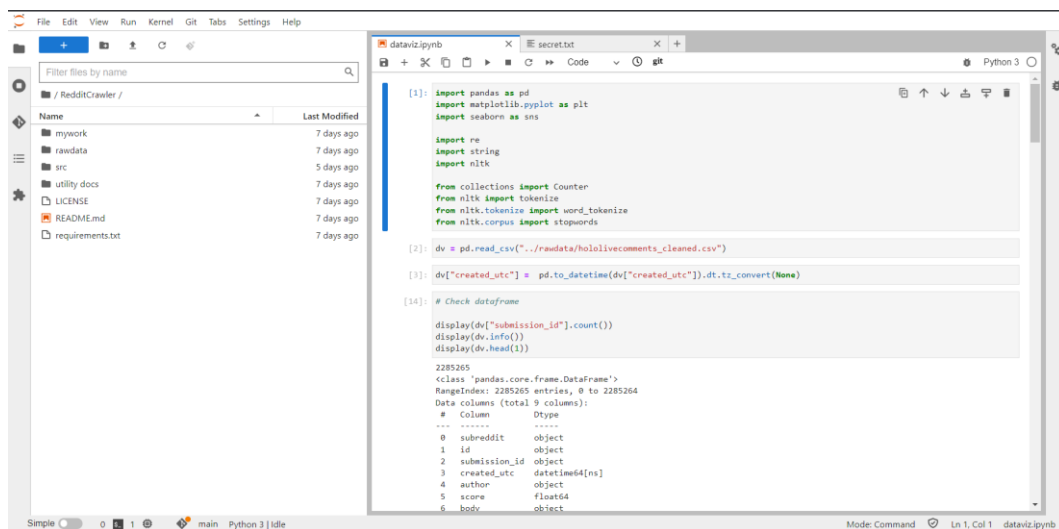
*Virtual machine* digunakan untuk menjalankan OS Ubuntu. Proses crawling/ekstraksi data dilakukan menggunakan python *environment* yang berjalan di atas OS Ubuntu. Penggunaan Ubuntu Virtual Machine dimaksudkan akan proses ekstraksi data yang memerlukan waktu lama dapat terus berjalan di sisi server sehingga tidak diperlukan sistem lokal untuk berjalan secara terus-menerus. Gambar IV-6 merupakan Ubuntu Virtual Machine yang digunakan untuk berjalannya proses ekstraksi data pada Tugas Akhir ini.



Gambar IV-6 Tampilan Utama OS Ubuntu

#### IV.2.1.2 Jupyter Notebook Server

Beberapa proses komputasi data yang berjumlah jutaan memerlukan waktu yang lama untuk mengeksekusinya. Oleh karena itu, diperlukan juga *notebook/workbench* yang dapat berjalan di sisi *server*. Google Cloud Platform mempunyai layanan bernama Vertex AI yang dapat difungsikan untuk pengolahan *big data* dan *machine learning*. Gambar IV-7 merupakan tampilan dari *notebook* yang berjalan di GCP.



Gambar IV-7 Tampilan Python Notebook di Server GCP

#### IV.2.2 NoMachine Remote Desktop

NoMachine adalah aplikasi perangkat lunak lintas platform berpaten untuk akses jarak jauh, berbagi desktop, desktop virtual, dan transfer file antar komputer. NoMachine dapat diinstal pada komputer dengan OS Windows, Mac, Linux, Raspberry Pi dan Linux ARM untuk memungkinkan pengguna mengakses desktop dari jarak jauh melalui jaringan. Pengguna dapat terhubung dari Windows, macOS, iOS, Android, Linux, Raspberry Pi, Linux ARM atau browser web. NoMachine pada Tugas Akhir ini, digunakan untuk menjalankan virtual machine GCP dengan koneksi yang sudah disediakan. Gambar IV-8 merupakan tampilan awal NoMachine yang menunjukkan pilihan sistem remote tersimpan.



Gambar IV-8 Tampilan Utama Aplikasi NoMachine

### IV.3 Pengujian Kemampuan *Crawling* Sistem

Pengujian ini dilakukan untuk melihat seberapa reliabel sistem dalam mengambil data kasar dari Reddit. Pengambilan data dilakukan dalam beberapa kondisi dan dibandingkan untuk mencari metode yang paling efektif dalam menjalankan skenario tersebut.

#### IV.3.1 Pengujian Lama Waktu Proses Berdasarkan *Network*

Pengujian ini bertujuan untuk mendapatkan perbandingan lama waktu yang dibutuhkan untuk menyelesaikan satu skenario kondisi yang akan diujikan. Kondisi yang diuji adalah jenis *network* yang digunakan oleh mesin dalam menjalankan fungsi ekstraksi data dari API Reddit. Pengujian ini menggunakan perintah yang sama, sehingga data yang didapatkan juga merupakan data yang identik. Ekstraksi data dilakukan dengan 10 *batch* dalam 3 *lap*. *Batch* merupakan limit yang ditetapkan dalam mengambil *submission* dalam satu waktu, sedangkan *lap* merupakan berapa kali pemanggilan *batch* dalam satu perintah. Dengan menggunakan nilai *batch* dan *lap* yang sama, diharapkan bahwa item yang didapatkan dari proses ekstraksi juga memiliki jumlah yang sama. Perintah terminal yang digunakan adalah.

```
python src/subreddit_downloader.py Hololive --batch-size 10 --laps 3 --reddit-id DnpX9tZ075idVbdUuDUGdg -
--reddit-secret 8-8kRY17k-I0KsHRh7ey-ZCQjks8vA --
reddit-username iammedesu --utc-before 1676946171
```

Tabel IV-5 Data Hasil Percobaan Pengujian Lama Waktu

<i>Internet Provider</i>	Proxy/VPN	Kecepatan unduh (Mb/detik)	Kecepatan ping (ms)	Waktu rata-rata setiap lap (menit)	Waktu total (menit)	Item yang didapat (row)
Biznet	-	82	11	Gagal	Gagal	Gagal
Biznet	ITB VPN	57	40	0,46	1,4	157
Biznet	Cloudflare 1.1.1.1	88	256	0,86	2,6	157
Firstmedia	ITB VPN	28	23	0,16	0,5	157
Firstmedia	Cloudflare 1.1.1.1	94	266	0,13	0,4	157
Eduroam	-	15	19	0,26	0,8	157
<i>Cloud network</i>	-	1100	8	0,03	0,1	157

Dari beberapa skenario pengujian di Tabel IV-5, didapatkan bahwa proses tercepat untuk perolehan 157 rows data adalah dengan menggunakan *cloud network*. Dapat pula disimpulkan bahwa *cloud network* mempunyai kecepatan unduh dan nilai ping



yang lebih baik dibandingkan dengan skenario pengujian network lainnya. Oleh karena itu, penggunaan *cloud network* akan diimplementasikan pada pengujian proses ekstraksi data selanjutnya.

#### IV.3.2 Pengujian Jumlah Item Berdasarkan Lama Waktu Proses

Pengujian ini juga bertujuan untuk mendapatkan perbandingan lama waktu yang dibutuhkan untuk menyelesaikan satu skenario kondisi yang akan diujikan. Pengujian ini dilakukan dengan *cloud network* seperti yang sudah dijelaskan pada bagian 3.1. Kondisi yang diuji adalah jumlah data yang dimintakan ke API Reddit. Pengujian ini menggunakan perintah yang berbeda untuk setiap command yang diberikan. Variasi yang digunakan dalam pengujian ini adalah jumlah *lap* dan jumlah *batch* scraping. Perintah terminal yang digunakan adalah sebagai berikut (dengan nilai x dan y yang bervariasi).

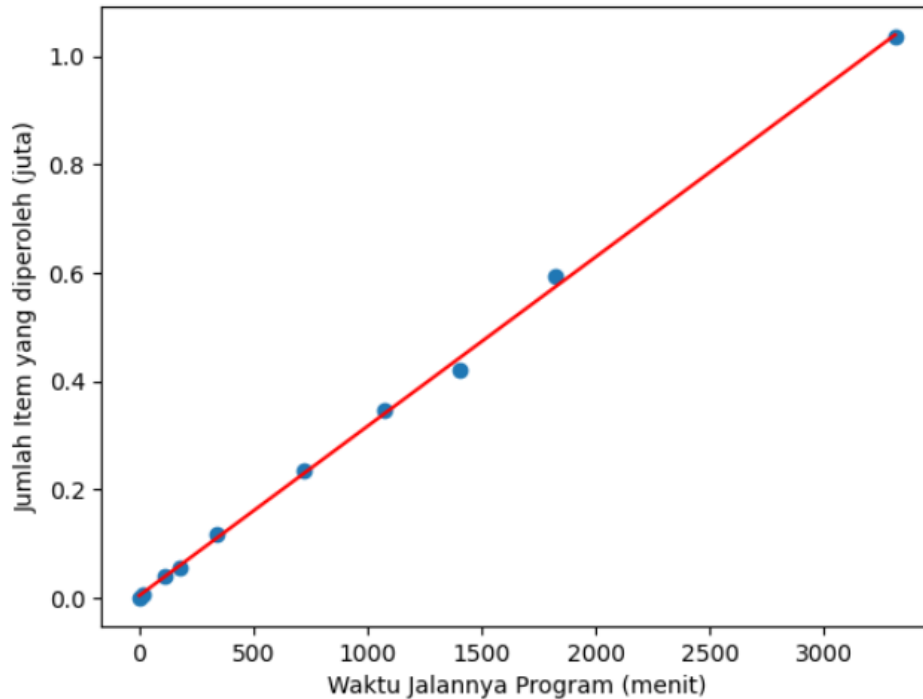
```
python src/subreddit_downloader.py Hololive --batch-size x --laps y --reddit-id DnpX9tZO75idVbdUuDUgdg --reddit-secret 8-8kRY17k-I0KsHRh7ey-ZCQjks8vA --reddit-username iammedesu --utc-before 1676946171
```

Tabel IV-6 Data Hasil Percobaan Pengujian Jumlah Item

<i>Internet Provider</i>	Proxy/VPN	Ukuran batch (x)	Jumlah lap (y)	Waktu rata-rata setiap lap (menit)	Waktu total (menit)	Item yang didapat
<i>Cloud network</i>	-	10	3	0,03	0,1	157

<i>Internet Provider</i>	Proxy/VPN	Ukuran batch (x)	Jumlah lap (y)	Waktu rata-rata setiap lap (menit)	Waktu total (menit)	Item yang didapat
<i>Cloud network</i>	-	1000	1	18,3	18,3	5957
<i>Cloud network</i>	-	1000	6	18,23	109,4	40486
<i>Cloud network</i>	-	1000	10	17,56	175,6	54697
<i>Cloud network</i>	-	1000	20	17,03	340,6	117885
<i>Cloud network</i>	-	1000	40	18,07	722,8	236340
<i>Cloud network</i>	-	1000	60	17,92	1075,2	345979
<i>Cloud network</i>	-	1000	80	17,6	1407,4	421088
<i>Cloud network</i>	-	1000	100	18,24	1824,7	592810
<i>Cloud network</i>	-	1000	177	18,7	3310,8	1035939

Dari Tabel IV-6, dapat diplot suatu grafik jumlah item yang didapatkan berdasarkan lama waktu satu proses ekstraksi tersebut berlangsung. Didapatkan pula garis regresi linier dari pemodelan grafik tersebut.



Gambar IV-9 Grafik Plot Total Item Berdasarkan Waktu

Gambar IV-9 merupakan grafik plot total item yang didapatkan dari Reddit berdasarkan waktu (menit) yang dihabiskan. Nilai persamaan regresi yang didapatkan adalah:

$$f(x) = 312,8x + 4093$$

Dengan  $f(x)$  adalah jumlah item yang akan didapatkan dan  $x$  adalah waktu yang ditetapkan. Asumsi bahwa ketika waktu yang berjalan adalah 0 menit ( $x = 0$ ), maka tidak ada data yang diperoleh ( $f(x) = 0$ ), sehingga intercept grafik regresi dapat juga diasumsikan berada di *origin* ( $f(0) = 0$ ). Sehingga nilai regresi menjadi.

$$f(x) = 312,8x$$

#### IV.3.3 Pengujian Ukuran Data Berdasarkan Jumlah Item yang diperoleh

Pengujian ini juga bertujuan untuk mendapatkan perbandingan ukuran data di penyimpanan pada suatu skenario kondisi yang akan diujikan. Pengujian dilakukan dengan *network* yang sama yaitu *cloud network* dan menggunakan variasi jumlah *lap* sehingga jumlah item yang diperoleh pun diharapkan berbeda. Perintah terminal yang digunakan adalah sebagai berikut (dengan nilai *x* yang bervariasi).

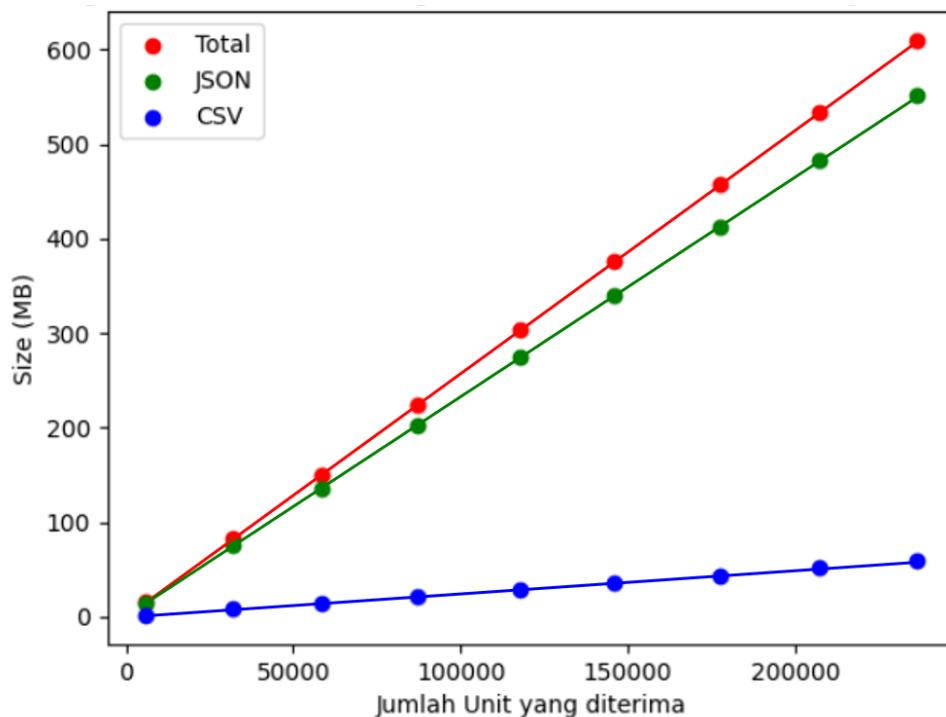
```
python src/subreddit_downloader.py Hololive --batch-size 1000 --laps x --reddit-id DnpX9tZ075idVbdUuDUGdg --reddit-secret 8-8kRY17k-I0KsHRh7ey-ZCQjks8vA --reddit-username iammedesu --utc-before 1676946171
```

Tabel IV-7 Data Hasil Percobaan Pengujian Ukuran Data

Jumlah lap (x)	Waktu total (menit)	Jumlah item	Ukuran JSON (MB)	Ukuran CSV (MB)	Ukuran total (MB)
1	17	5956	13,9	1,5	15,4
5	89,7	32128	75,2	8,2	83,4
10	170,8	58256	135,8	14,4	150,2
15	259,3	86960	202,3	21,5	223,8
20	348,4	117877	274,4	29,2	303,6
25	428,3	146100	340,4	36,2	376,6
30	521,2	177287	413	43,8	456,8
35	623,9	207165	482,7	51,2	533,9

Jumlah lap (x)	Waktu total (menit)	Jumlah item	Ukuran JSON (MB)	Ukuran CSV (MB)	Ukuran total (MB)
40	717,6	236340	551,2	58,5	609,7

Dari Tabel IV-7, dapat dibuatkan sebuah grafik ukuran data JSON, ukuran data CSV, maupun ukuran data total (gabungan JSON dan CSV) berdasarkan jumlah item yang didapatkan dari proses ekstraksi data. Dapat dihitung pula formula regresi linier dari grafik tersebut untuk memprediksikan jumlah kapasitas penyimpanan yang diperlukan untuk melakukan ekstraksi data dari sekian item yang ditargetkan.



Gambar IV-10 Grafik Plot Ukuran data Berdasarkan Jumlah Unit

Dari Gambar IV-10, dapat dilihat bahwa *plotting* ukuran penyimpanan yang dibutuhkan untuk menyimpan hasil ekstraksi data berdasarkan jumlah unit yang diterima. Hasil percobaan yang digunakan untuk menentukan prediksi ukuran

penyimpanan hasil ekstraksi data adalah ukuran total data gabungan CSV dan JSON. Didapatkan juga persamaan regresi jumlah unit yang diterima terhadap ukuran total:

$$g(x) = 0,0026x + 0,0502$$

Dengan  $g(x)$  adalah ukuran (size) dalam megabyte (MB) yang dibutuhkan untuk menyimpan data berdasarkan jumlah unit yang diterima ( $x$ ) dari proses ekstraksi data dari Reddit. Asumsi bahwa ketika jumlah unit yang diterima adalah 0 ( $x = 0$ ), maka ukuran data juga 0 MB ( $g(x) = 0$ ), sehingga intercept grafik regresi dapat juga diasumsikan berada di *origin* ( $g(0) = 0$ ). Sehingga nilai regresi menjadi.

$$g(x) = 0,0026x$$

Sebagai contoh, ketika menargetkan ada sekitar satu juta unit data yang diekstraksi dari Reddit, maka perkiraan ukuran penyimpanan yang dibutuhkan untuk menampung file JSON dan CSV adalah 2600 MB (megabyte).

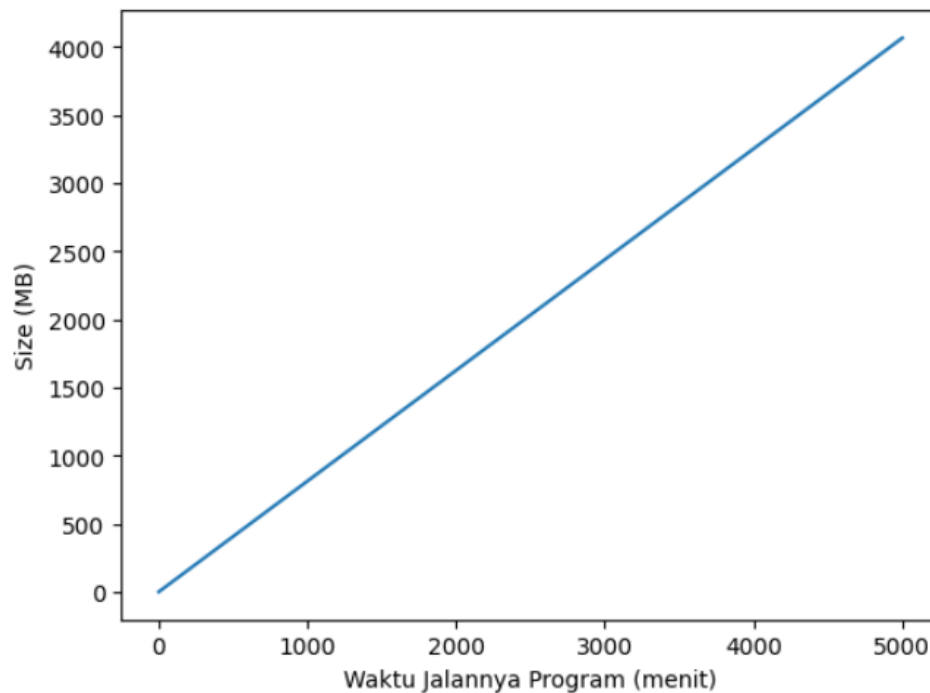
Dari persamaan regresi  $f(x)$  dari bagian IV.3.2 dan persamaan regresi  $g(x)$ , dapat dibuat juga persamaan regresi  $h(x)$  untuk memberikan keterhubungan antara waktu proses ekstraksi berjalan (dalam menit) terhadap ukuran penyimpanan yang dibutuhkan (dalam MB). Persamaan regresi linier  $g(x)$  dapat ditentukan dengan cara sebagai berikut.

$$h(x) = f(g(x)) = 312,8(g(x))$$

$$h(x) = 312,8(0,0026x)$$

$$h(x) = 0,81328x$$

Dari persamaan garis regresi tersebut, dapat digambarkan grafik plot (Gambar IV-11) keterhubungan lama waktu proses ekstraksi (dalam menit) terhadap ukuran penyimpanan yang dibutuhkan (dalam MB).



Gambar IV-11 Grafik Plot Ukuran Data Berdasarkan Waktu

Dapat diperkirakan juga, ketika ingin melakukan proses ekstraksi selama satu minggu penuh (168 jam), maka total ukuran penyimpanan minimal yang dibutuhkan selama proses adalah 8197,86 MB (megabyte).

#### IV.3.4 Pengujian Konversi JSON ke CSV dan Database

Untuk menunjukkan format data yang paling optimal untuk proses CTI meliputi pencarian data maupun pemrosesan analisis data, diperlukan pengujian terhadap beberapa format umum untuk suatu kumpulan data. Format umum yang dipilih pada pengujian ini adalah CSV (*Comma Separated Value*) dan *database* (dalam SQLite). Kedua jenis format ini merupakan yang paling umum digunakan dalam merepresentasikan suatu kumpulan data. Pengujian akan dilakukan dengan melakukan perbandingan konversi dari format JSON (format *default* dari Reddit API) ke dalam bentuk CSV dan *database*.

Tabel IV-8 Konversi JSON ke CSV dan *Database*

Jumlah baris	CSV <i>time</i> (detik)	Database <i>time</i> (detik)
--------------	-------------------------	------------------------------

3	0.00400090217590332	0.23458552360534668
5	0.003999948501586914	0.23686671257019043
10	0.00500178337097168	0.22804474830627441
25	0.013000011444091797	0.23559308052062988
50	0.013001680374145508	0.27994370460510254
80	0.01099967956542968	0.2850363254547119
200	0.015003204345703125	0.2646205425262451

Dari Tabel IV-8, dapat disimpulkan bahwa rata-rata waktu konversi dari JSON ke CSV relatif lebih singkat dibandingkan dengan konversi JSON ke *database*. Hal ini dapat memangkas waktu untuk pemilihan *field/parsing data* dari hasil *crawling data* yang masih berbentuk *raw*. Selain itu, walaupun SQL merupakan bahasa pemrograman yang lebih efektif dalam melakukan *query* data, penggunaan CSV memiliki kapabilitas yang lebih dalam analisis suatu tabel *database*. Sebuah library bernama Pandas cukup ideal untuk *user* yang perlu menggunakan perintah sederhana dan menganalisis data terstruktur.

#### IV.4 Persiapan Data

Untuk menunjang proses analisis data, terdapat suatu prosedur untuk memastikan kebenaran, konsistensi, dan kegunaan suatu data yang ada dalam *dataset*. Tugas Akhir ini menggunakan subreddit “Hololive” sebagai subyek pengujian.

##### IV.4.1 Penghapusan Baris Ganda

Penghapusan baris ganda atau duplikat pada *dataset* sangat penting dalam memastikan keakuratan analisis data dan menghindari adanya bias atau kesalahan dalam hasil analisis. Oleh karena itu, penghapusan baris duplikat harus dilakukan dengan hati-hati dan berdasarkan kriteria tertentu, seperti mempertimbangkan kolom apa yang harus diperiksa untuk mendeteksi duplikasi dan memastikan bahwa data yang diperlakukan sama persis dalam setiap baris. Dengan demikian, penghapusan baris duplikat dapat memastikan bahwa analisis data yang dilakukan



dilakukan pada *dataset* yang berkualitas tinggi dan akurat. Digunakanlah kode program sebagai berikut untuk menghapus baris yang duplikat.

```
df.drop_duplicates(inplace=True)
```

Fungsi `drop_duplicates()` digunakan untuk menghapus baris duplikat pada *dataframe*. Argumen `inplace=True` digunakan untuk mengubah *dataframe* asli, yaitu menghapus baris duplikat pada *dataframe* yang diberikan.

#### IV.4.2 Manajemen Data Kosong

Dalam tabel *dataset* hasil ekstraksi data dari Reddit, selalu terdapat kemungkinan adanya data yang kosong maupun dengan kondisi tertentu. Oleh karena itu, diperlukan sebuah metode untuk mengelola data kosong, seperti penghapusan atau imputasi data.

##### IV.4.2.1 Penggantian Data Tertentu dengan Nilai Kosong

Pada kolom *body* atau kolom yang berisikan komentar di dalam *dataset*, terdapat beberapa data yang berisikan nilai string “[removed]”. Hal ini dapat diasumsikan bahwa komentar telah dihapus oleh *Author* maupun moderator. Oleh karena itu, data ini akan diubah menjadi nilai kosong sehingga tidak membuat bias pada analisis maupun visualisasi data dan memudahkan langkah selanjutnya untuk dihapus dari *dataframe*. Digunakanlah kode program sebagai berikut.

```
df['body'] = df['body'].replace('[removed]', np.nan)
```

Potongan kode di atas merupakan kode Python yang mengganti setiap nilai string “[removed]” dalam kolom *body dataframe* dengan nilai NaN menggunakan metode `replace()` dari *pandas library*. Nilai NaN (Not a Number) digunakan untuk mewakili nilai kosong atau tidak valid dalam *dataframe*.

#### IV.4.2.2 Pengisian Nilai Kosong Menjadi Data Tertentu

Pada kolom *Author* atau kolom yang berisikan *username* Reddit yang menuliskan komentar, terdapat beberapa data yang bernilai null atau data kosong. Hal ini dapat diasumsikan bahwa *Author* atau *user* yang berkomentar membuat visibilitas akunnya terbatas. Maka dari itu, diubahlah data kosong di kolom *Author* menjadi nilai string “NaN”, dengan asumsi bahwa *username* dengan nilai string tersebut tidak ada di Reddit atau sudah tidak aktif dan tidak memberikan *post* ke subreddit yang dianalisis. Digunakan kode program sebagai berikut untuk mengganti nilai kosong menjadi nilai tertentu.

```
df['Author'] = df['Author'].fillna('NaN')
```

Potongan kode di atas merupakan kode Python yang mengisi nilai kosong (NaN) pada kolom *Author* dalam *dataframe* dengan nilai string ‘NaN’ menggunakan metode `fillna()` dari *pandas library*.

#### IV.4.3 Pembersihan Data

Terdapat beberapa kemungkinan data yang masih bernilai kosong dikarenakan kegagalan maupun error dalam proses ekstraksi data menggunakan PRAW dan data kosong yang tersisa pada langkah sebelumnya. Untuk mengatasinya, seluruh data yang masih bernilai kosong atau NA akan dihapus dari *dataframe*. Digunakan kode program sebagai berikut.

```
df.dropna(axis=0, how='any', subset=None, inplace=True)
```

Kode tersebut menggunakan fungsi `dropna()` untuk menghapus baris pada *dataframe* yang mengandung nilai kosong (NaN) atau hilang. Argumen `axis=0` menunjukkan bahwa fungsi harus menghapus baris yang mengandung nilai kosong, sementara `how='any'` mengindikasikan bahwa setidaknya satu nilai kosong dalam suatu baris akan menyebabkan baris tersebut dihapus. Argumen `subset=None` digunakan untuk menunjukkan bahwa semua kolom dalam *dataframe* harus

diperiksa untuk nilai kosong, dan argumen `inplace=True` digunakan untuk mengubah *dataframe* asli.

#### **IV.4.4 Manajemen *Stopwords* dan *Irrelevant Items***

*Stopwords* adalah kata-kata yang tidak memuat informasi yang signifikan dalam dokumen atau kalimat dan dapat diabaikan dalam analisis. Hal ini dapat membantu mempercepat analisis teks dan mengurangi keberisian dalam model. Beberapa *framework* pemrosesan bahasa alami, seperti NLTK (Natural Language Toolkit), menyediakan daftar *stopwords* bawaan untuk berbagai bahasa. Program berikut merupakan algoritma penghapusan *stopwords* dan beberapa bagian yang tidak memuat informasi signifikan pada *dataframe*.

```

nltk.download('punkt')
nltk.download('stopwords')

STOP_WORDS = stopwords.words()

EMOJI_PATTERN = re.compile("[
                                u\"\U0001F600-\U0001F64F" # emoticons
                                u\"\U0001F300-\U0001F5FF" # symbols &
                                pictographs
                                u\"\U0001F680-\U0001F6FF" # transport &
                                map symbols
                                u\"\U0001F1E0-\U0001F1FF" # flags (iOS)
                                u\"\U00002702-\U000027B0"
                                u\"\U000024C2-\U0001F251"
                                "]" +, flags=re.UNICODE)

def cleaning(text):
    text = text.lower()
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('%s' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('[''"]', '', text)
    text = EMOJI_PATTERN.sub(r'', text)
    text_tokens = word_tokenize(text)
    tokens_without_sw = [word for word in text_tokens if not word in
STOP_WORDS]
    filtered_sentence = (" ").join(tokens_without_sw)
    text = filtered_sentence

    return text

```

Kode di atas digunakan untuk melakukan pra-pemrosesan data teks dalam sebuah fungsi bernama `cleaning()`. Pra-pemrosesan data teks ini bertujuan untuk membersihkan teks dari karakter yang tidak relevan atau mengganggu seperti tanda baca, emotikon, dan *stop words* yang tidak memiliki arti dalam analisis teks. Langkah-langkah yang dilakukan dalam fungsi `cleaning()` adalah. Pertama, teks dikonversi ke huruf kecil untuk konsistensi. Kemudian, URL dan karakter khusus dihapus dari teks menggunakan bantuan *regular expression*. Setelah itu, tanda baca dan karakter khusus lainnya juga dihapus dari teks. Teks kemudian di tokenisasi menggunakan `word_tokenize()` untuk memisahkan teks menjadi kata-kata. *Stop words* kemudian dihapus dari teks dengan membandingkan setiap kata dalam teks

dengan daftar *stop words* yang diambil dari *corpus stop words* di NLTK. Kata-kata yang sudah dihapus *stop words* kemudian digabungkan kembali menjadi sebuah kalimat tanpa *stop words*. Selain itu, kode juga menggunakan `re.compile()` dan pola *regular expression* untuk mengenali emotikon dan karakter khusus tertentu yang tidak diinginkan untuk menghapusnya. Pra-pemrosesan data teks sangat penting dalam analisis teks karena dapat meningkatkan kualitas data dan konsistensi data. Dalam kasus ini, fungsi `cleaning()` akan membantu menghapus informasi yang tidak relevan dan meningkatkan kualitas teks dalam *dataset* sehingga dapat digunakan untuk analisis teks yang lebih akurat.

## **IV.5 Visualisasi Data**

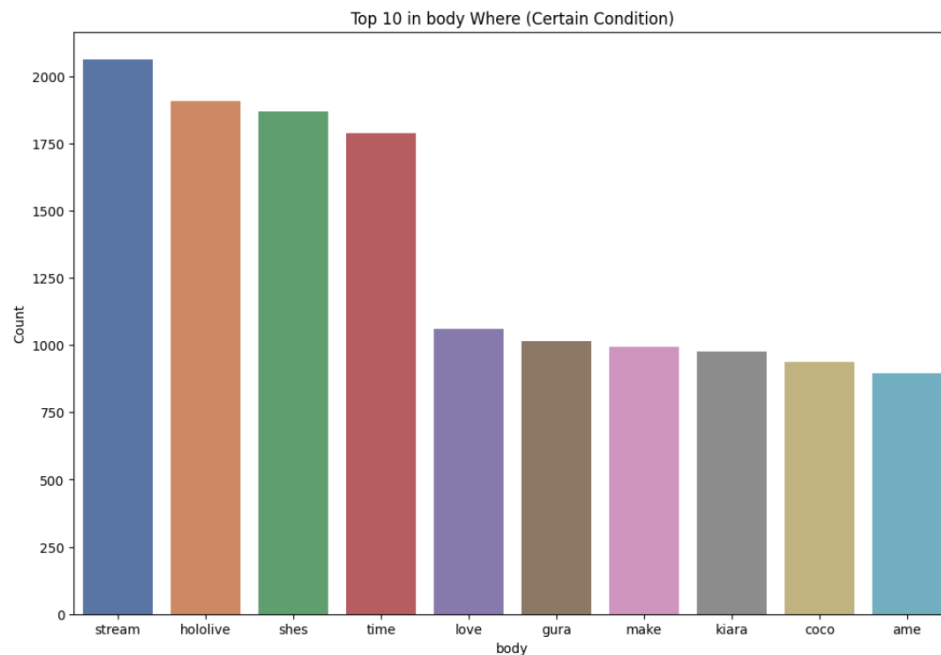
Visualisasi data memiliki peran yang sangat penting dalam analisis *cyber threat intelligence*, karena dapat membantu memahami data secara lebih baik dan mengungkap pola atau tren yang sulit terlihat dari data mentah yang ada. Dalam analisis, visualisasi data digunakan untuk mempresentasikan data dengan cara yang mudah dimengerti, sehingga *user*/pengguna dapat mengambil kesimpulan atau membuat keputusan berdasarkan data tersebut.

### **IV.5.1 Informasi yang Dapat diambil dari Data Mentah Hasil Ekstraksi**

Dari data hasil ekstraksi, dapat diberikan beberapa model visualisasi yang dapat memberikan informasi maupun insight yang berguna dalam analisis *cyber threat intelligence*. Contoh dibawah merupakan visualisasi dari data hasil ekstraksi subyek pengujian yang juga digunakan pada beberapa bab sebelumnya.

#### **IV.5.1.1 Kata yang Paling Banyak Muncul**

Berikut Gambar IV-12 di bawah merupakan hasil *plotting* grafik bar untuk kata yang paling banyak muncul dalam *dataframe*. Kata paling banyak muncul dibatasi dengan komentar (body) yang memiliki *score* lebih dari 250.



Gambar IV-12 Grafik Bar Kata Paling Banyak Muncul

Untuk melakukan *plotting* grafik bar di atas diperlukan kode sebagai berikut.

```
tokenSpace = tokenize.WhitespaceTokenizer()

def counter(text, columnText, quantity):
    allWords = ' '.join([text for text in
text[columnText].astype('str')])
    tokenPhrase = tokenSpace.tokenize(allWords)
    frequency = nltk.FreqDist(tokenPhrase)
    dfFrequency = pd.dataframe({columnText: list(frequency.keys()),
"Frequency": list(frequency.values())})

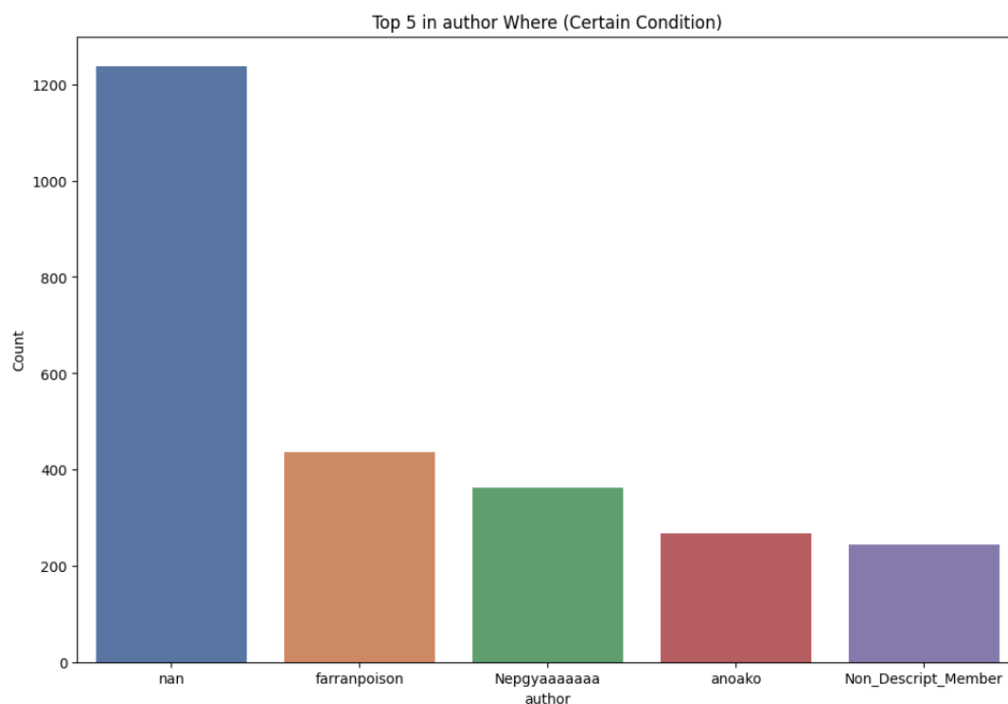
    dfFrequency = dfFrequency.nlargest(columns = "Frequency", n =
quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = dfFrequency, x = columnText, y =
"Frequency", palette="deep")
    ax.set(ylabel = "Count")
    ax.set_title('Top {} in {} Where (Certain
Condition)'.format(quantity, columnText))
    plt.xticks(rotation='horizontal')
    plt.show()

counter(dv[dv["score"] >= 250], "body", 10)
```

Sebuah fungsi bernama “counter” yang digunakan untuk menghitung frekuensi kata-kata pada sebuah kolom teks dari sebuah *dataset*. Fungsi ini menerima tiga argumen, yaitu “text” yang merepresentasikan *dataset* yang akan dihitung, “columnText” yang merepresentasikan kolom teks yang ingin dihitung, dan “quantity” yang merepresentasikan jumlah kata yang ingin ditampilkan.

#### IV.5.1.2 *Author* yang Paling Banyak Berkomentar

Berikut Gambar IV-13 di bawah merupakan hasil *plotting* grafik bar untuk *Author* yang paling banyak berkomentar di dalam *dataframe*. *Author* paling banyak berkomentar dibatasi dengan score yang lebih dari 250. Terdapat kasus khusus di nama *Author* “nan” yang sudah dibahas pada bagian IV.4.2.2. Nama “nan” menjelaskan pada field *Author* yang pada *dataframe* merupakan nilai kosong atau diasumsikan *Author* tersebut tidak membagikan *username*-nya kepada publik. Oleh karena itu, pada analisis selanjutnya, nama “nan” dapat diabaikan maupun digunakan sebagai pertimbangan lain untuk kasus nama *user* yang anonymous.



Gambar IV-13 Grafik Bar *Author* Paling Banyak Berkomentar

Untuk melakukan *plotting* grafik bar di atas diperlukan kode sebagai berikut.

```
tokenSpace = tokenize.WhitespaceTokenizer()

def counter(text, columnText, quantity):
    allWords = ' '.join([text for text in
text[columnText].astype('str')])
    tokenPhrase = tokenSpace.tokenize(allWords)
    frequency = nltk.FreqDist(tokenPhrase)
    dfFrequency = pd.dataframe({columnText: list(frequency.keys()),
"Frequency": list(frequency.values())})

    dfFrequency = dfFrequency.nlargest(columns = "Frequency", n =
quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = dfFrequency, x = columnText, y =
"Frequency", palette="deep")
    ax.set(ylabel = "Count")
    ax.set_title('Top {} in {} Where (Certain
Condition)'.format(quantity, columnText))
    plt.xticks(rotation='horizontal')
    plt.show()

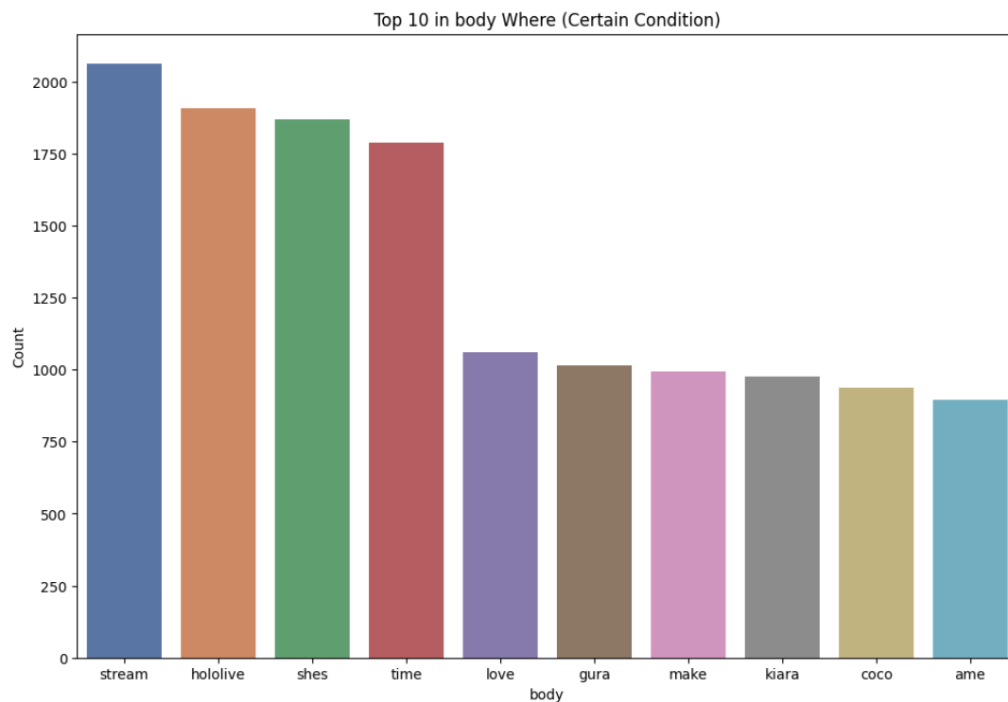
counter(dv[dv["score"] >= 250], "Author", 5)
```

Fungsi yang digunakan sama seperti pada bagian IV.5.1.1. Akan tetapi, menggunakan argumen pemanggilan fungsi yang berbeda yaitu memilih kolom tabel “*Author*” untuk menunjukkan akumulasi kata terbanyak di kolom tersebut. Asumsi yang diambil adalah setiap nama *user* Reddit merupakan sebuah kata yang unik dan berbeda satu sama lain.

#### **IV.5.1.3 Kata yang Paling Banyak digunakan Oleh Seorang *Author***

Berikut Gambar IV-14 di bawah merupakan hasil *plotting* grafik bar untuk akumulasi kata yang paling sering muncul pada *Author* yang sama. Grafik di bawah menggunakan contoh pada *Author* yang paling banyak berkomentar.





Gambar IV-14 Kata Paling Banyak digunakan Oleh *Author* Tertentu

Untuk melakukan *plotting* grafik bar di atas diperlukan kode sebagai berikut.

```
tokenSpace = tokenize.WhitespaceTokenizer()

def counter(text, columnText, quantity):
    allWords = ' '.join([text for text in
text[columnText].astype('str')])
    tokenPhrase = tokenSpace.tokenize(allWords)
    frequency = nltk.FreqDist(tokenPhrase)
    dfFrequency = pd.dataframe({columnText: list(frequency.keys()),
"Frequency": list(frequency.values())})

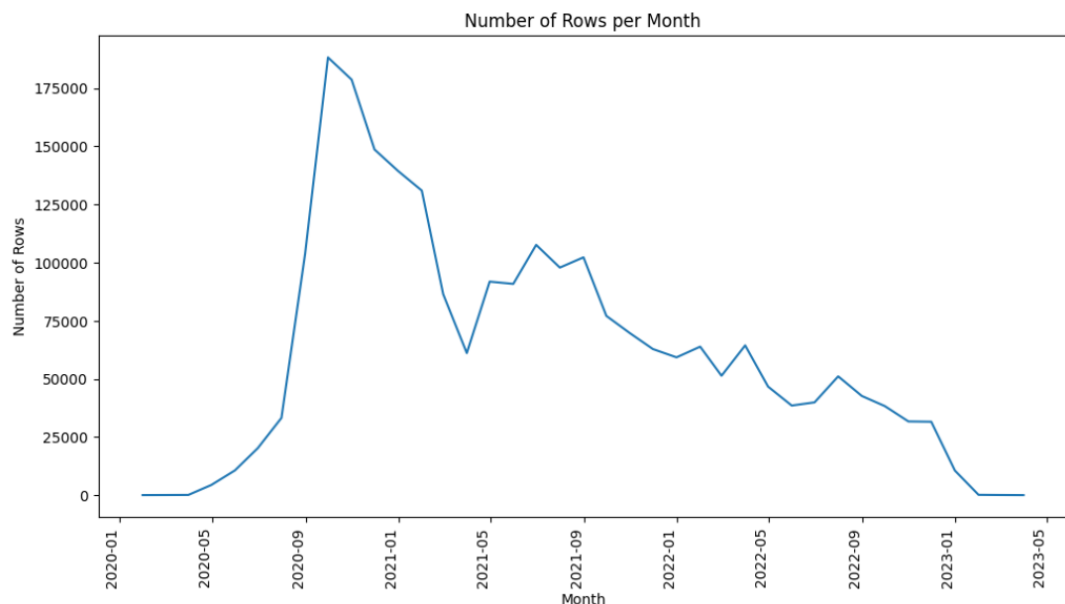
    dfFrequency = dfFrequency.nlargest(columns = "Frequency", n =
quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = dfFrequency, x = columnText, y =
"Frequency", palette="deep")
    ax.set_ylabel = "Count"
    ax.set_title('Top {} in {} Where (Certain
Condition)'.format(quantity, columnText))
    plt.xticks(rotation='horizontal')
    plt.show()

counter(dv[dv["Author"] == "farranpoison"], "body", 10)
```

Fungsi yang digunakan juga sama seperti pada bagian IV.5.1.1. Akan tetapi, tabel yang digunakan sedikit dimodifikasi sehingga hanya muncul data yang dibatasi untuk *Author* hanya nama tertentu saja (pada kasus di atas, nama *Author* “farranpoison” atau *user* yang paling banyak berkomentar). Modifikasi dilakukan pada argumen nama tabel yang dibentuk sedemikian rupa sehingga hanya menunjukkan bahwa hanya *Author* tersebut yang dimunculkan.

#### IV.5.1.4 Jumlah Komentar Per Bulan

Berikut Gambar IV-15 di bawah merupakan hasil *plotting* grafik garis untuk jumlah komentar yang muncul per bulan.



Gambar IV-15 Jumlah Komentar Per Bulan

Untuk melakukan *plotting* grafik garis di atas diperlukan kode sebagai berikut.

```
dv1 = dv.copy()

dv1.set_index('created_utc', inplace=True)

monthly_counts = dv1.resample('M').size()

plt.subplots(figsize=(12,6))
plt.plot(monthly_counts.index, monthly_counts.values)
plt.title('Number of Rows per Month')
plt.xlabel('Month')
plt.ylabel('Number of Rows')

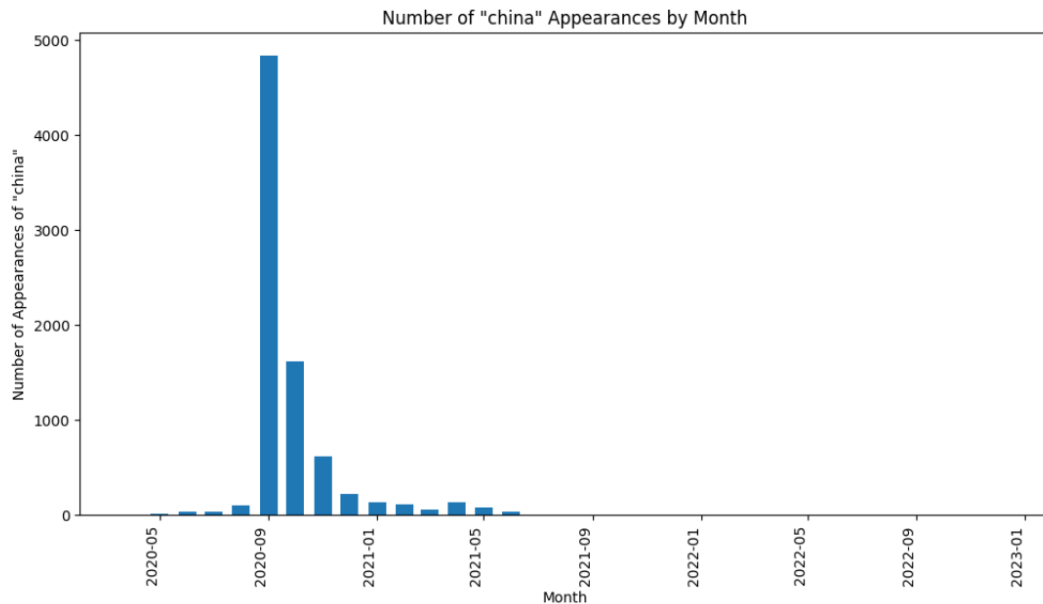
plt.xticks(rotation=90, ha='right')

plt.show()
```

Kode tersebut akan menghasilkan sebuah grafik garis yang menunjukkan jumlah baris dalam *dataset* per bulan. Pertama-tama, *dataframe* *dv* akan disalin ke dalam *dv1* agar data asli tidak berubah. Kemudian, kolom *created\_utc* akan dijadikan indeks. Selanjutnya, data akan dikelompokkan berdasarkan bulan dan dihitung jumlah baris per bulan menggunakan fungsi *resample* dan *size*. Hasilnya akan digambarkan dalam bentuk grafik garis dengan sumbu x menunjukkan bulan dan sumbu y menunjukkan jumlah baris.

#### **IV.5.1.5 Jumlah Kemunculan Kata Tertentu Per Bulan**

Berikut Gambar IV-16 di bawah merupakan hasil *plotting* grafik bar untuk jumlah akumulasi kemunculan kata tertentu pada komentar per bulan.



Gambar IV-16 Grafik Jumlah Kemunculan Kata Tertentu Per Bulan

Untuk melakukan *plotting* grafik bar di atas diperlukan kode sebagai berikut.

```
def wordappear_permonth(word):
    dv1 = dv.copy()

    dv1['body'] = dv1['body'].astype(str)

    dv1['contains_word'] = dv1['body'].apply(lambda x: True if
re.search(r'\b{}\b'.format(word), x, re.IGNORECASE) else False)

    word_counts = dv1[dv1['contains_word'] ==
True].groupby(dv1['created_utc'].dt.to_period('M')).size()

    word_counts.index = word_counts.index.to_timestamp()

    fig, ax = plt.subplots(figsize=(12,6))
    ax.bar(word_counts.index, word_counts, width=20)
    ax.set_xlabel('Month')
    ax.set_ylabel('Number of Appearances of "{}".format(word))
    ax.set_title('Number of "{}" Appearances by Month'.format(word))

    plt.xticks(rotation=90, ha='right')

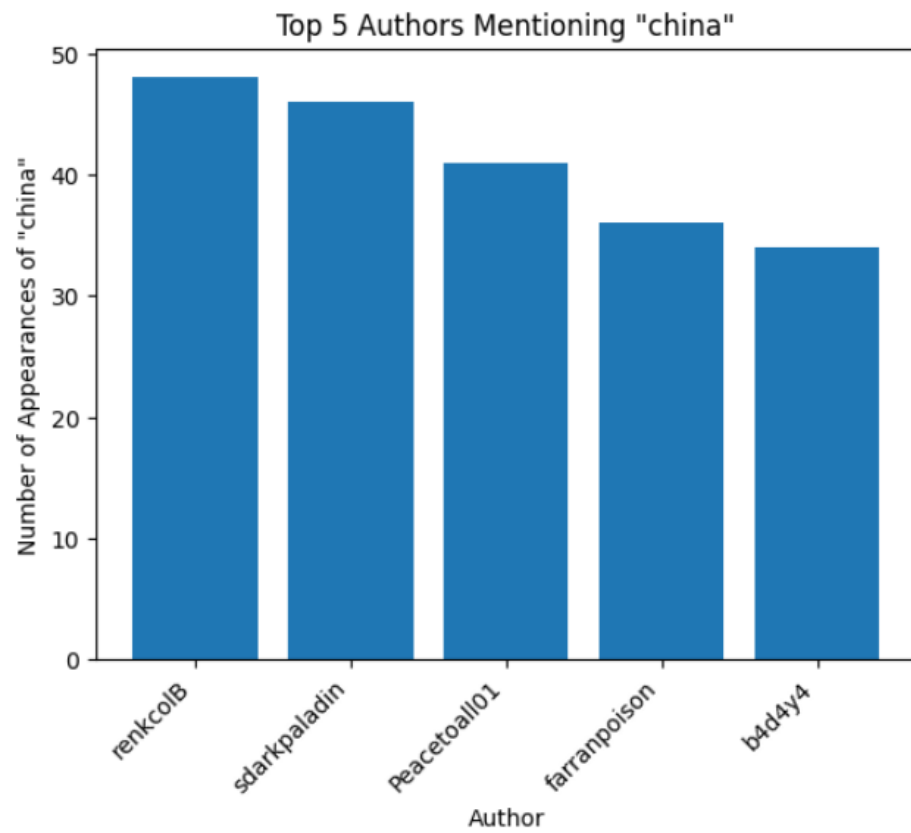
    plt.show()

wordappear_permonth("china")
```

Kode di atas merupakan sebuah fungsi `wordappear_permonth` yang menerima satu parameter input berupa “word”, yang akan digunakan untuk mencari berapa kali kata tersebut muncul dalam kolom body dari *dataframe* `dv`. Pertama-tama, fungsi akan membuat salinan *dataframe* `dv` menggunakan metode `copy()`, kemudian kolom body dari salinan *dataset* tersebut diubah menjadi string menggunakan metode `astype(str)` agar pencarian dengan regular expression dapat berfungsi dengan benar. Selanjutnya, fungsi akan membuat kolom baru bernama `contains_word` yang berisi nilai boolean yang menunjukkan apakah kata yang dicari ada dalam atribut body suatu baris atau tidak, dengan menggunakan *regular expression* `re.search` dengan parameter `\b{}\b` untuk mencari kata secara keseluruhan (whole word). Kemudian *dataframe* akan di-grup berdasarkan tahun dan bulan dari atribut `created_utc`, dan dihitung jumlah kemunculan kata pada setiap grup. Setelah itu, fungsi akan mengonversi objek `Period` ke `Timestamp`, dan menggambar grafik batang dengan menggunakan metode `bar()` dari `matplotlib`.

#### **IV.5.1.6 Author yang Paling Banyak Membicarakan Kata Tertentu**

Berikut Gambar IV-17 di bawah merupakan hasil *plotting* grafik bar untuk peringkat *Author* yang membicarakan kata tertentu dalam komentarnya berdasarkan akumulasi penggunaan kata tersebut.



Gambar IV-17 Grafik *Author* yang Membicarakan Kata Tertentu

Untuk melakukan *plotting* grafik bar di atas diperlukan kode sebagai berikut.

```

def top_Authormention(word, quantity):
    dv1 = dv.copy()

    dv1['body'] = dv1['body'].astype(str)

    dv1['contains_word'] = dv1['body'].apply(lambda x: True if
re.search(r'\b{}\b'.format(word), x, re.IGNORECASE) else False)

    word_counts = dv1[dv1['contains_word'] ==
True].groupby('Author').size().sort_values(ascending=False)[:quantity
]

    fig, ax = plt.subplots()
    ax.bar(word_counts.index, word_counts)
    ax.set_xlabel('Author')
    ax.set_ylabel('Number of Appearances of "{}".format(word))
    ax.set_title('Top {} Authors Mentioning "{}".format(quantity,
word))

    plt.xticks(rotation=45, ha='right')

    plt.show()

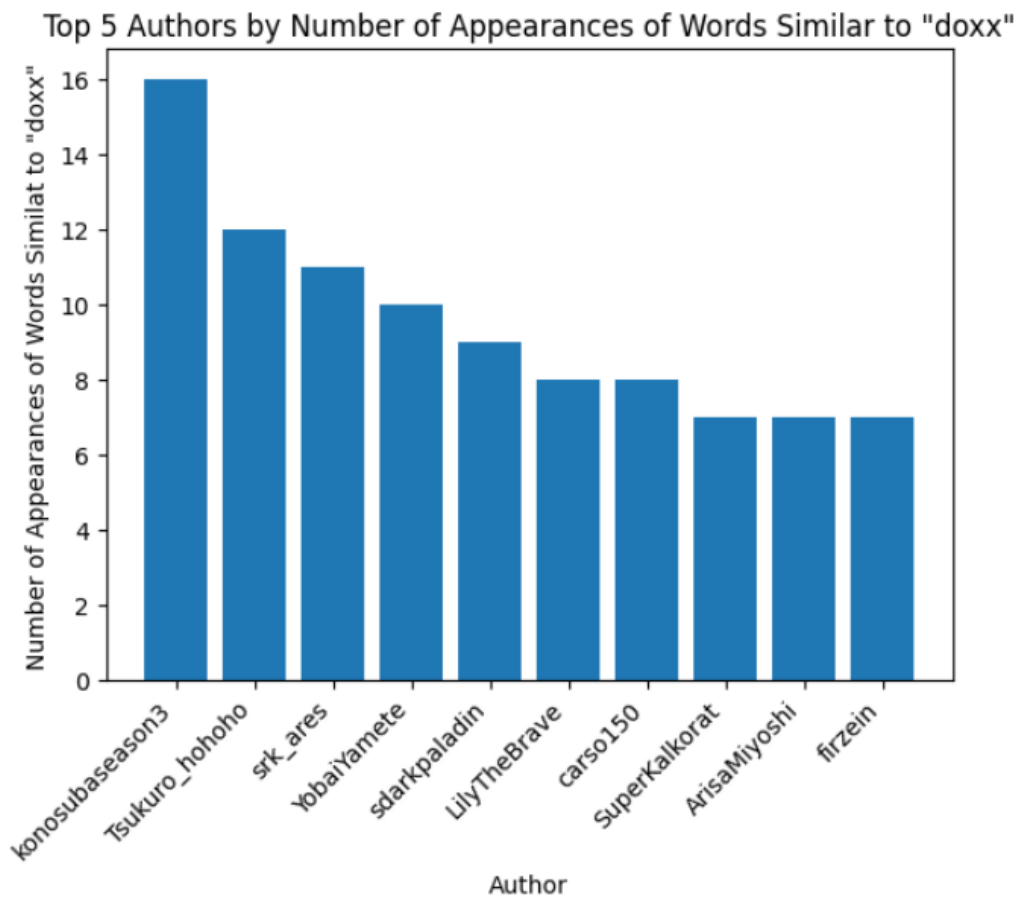
top_Authormention("china", 5)

```

Fungsi `top_Authormention` akan menampilkan grafik bar yang menunjukkan jumlah kemunculan kata yang dicari pada kolom body dan disebutkan oleh penulis pada kolom *Author* sebanyak `quantity` teratas. Fungsi ini akan menghasilkan grafik batang dengan sumbu x menunjukkan nama penulis dan sumbu y menunjukkan jumlah kemunculan kata tersebut oleh penulis.

#### IV.5.1.7 *Author* yang Paling Banyak Membicarakan Klaster Kata Tertentu

Berikut Gambar IV-18 di bawah merupakan hasil *plotting* grafik bar untuk peringkat *Author* yang membicarakan klaster atau kumpulan kata tertentu dalam komentarnya. Klaster kata adalah teknik pengelompokan kata atau istilah berdasarkan kemiripan makna, dengan tujuan untuk mempermudah analisis data teks. Pengumpulan klaster dapat dilakukan sesuai kebutuhan pengguna maupun dilakukan otomatisasi menggunakan algoritma pengelompokan data, seperti algoritma k-means atau *hierarchical clustering*, untuk menemukan kelompok-kelompok kata yang sering muncul bersama dalam teks.



Gambar IV-18 Grafik *Author* yang Membicarakan Topik Tertentu  
Untuk melakukan *plotting* grafik bar di atas diperlukan kode sebagai berikut.



```

def top_Authormentionlistofwords(those_words, quantity):
    dv1 = dv.copy()

    dv1['body'] = dv1['body'].astype(str)

    dv1['contains_those_words'] = dv1['body'].apply(lambda x: True if
any(re.search(r'\b{}\b'.format(word), x, re.IGNORECASE) for word in
those_words) else False)

    those_words_counts = dv1[dv1['contains_those_words'] ==
True].groupby('Author').size().sort_values(ascending=False)[:10]

    fig, ax = plt.subplots()
    ax.bar(those_words_counts.index, those_words_counts)
    ax.set_xlabel('Author')
    ax.set_ylabel('Number of Appearances of Words Similat to
"{}".format(those_words[0]))
    ax.set_title('Top {} Authors by Number of Appearances of Words
Similar to "{}".format(quantity, those_words[0]))
    plt.xticks(rotation=45, ha='right')
    plt.show()

those_words = ['doxx', 'harass', 'threat']
top_Authormentionlistofwords(those_words, 5)

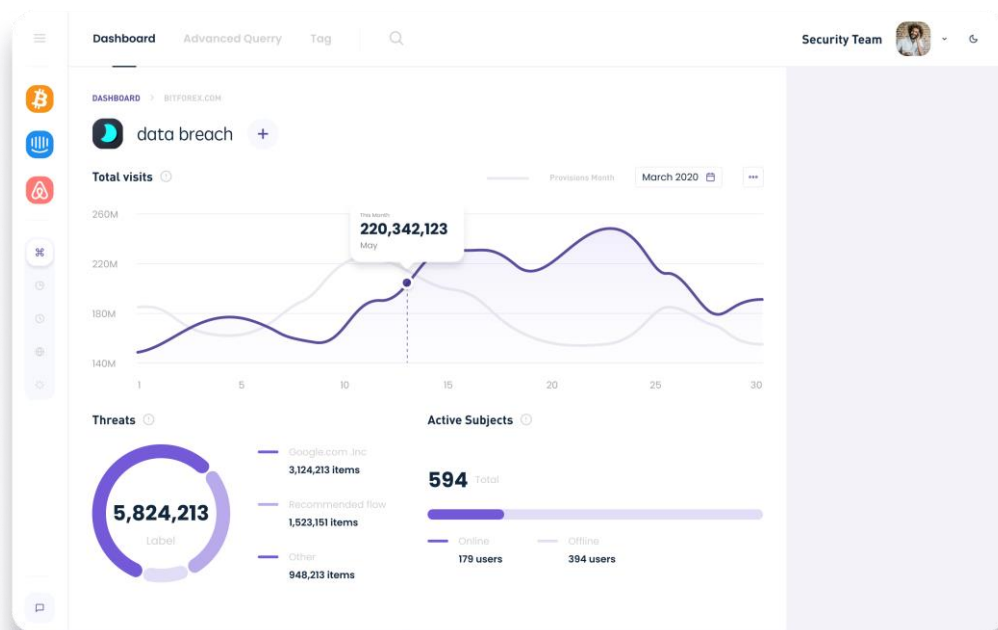
```

Kode tersebut terdapat sebuah fungsi yang bernama `top_Authormentionlistofwords`, yang menerima dua argumen yaitu `those_words` dan `quantity`. `those_words` berupa sebuah list of string yang berisi kumpulan kata-kata/klaster kata yang ingin dicari kemunculannya pada body dari data. Sedangkan `quantity` merupakan sebuah *integer* yang menentukan berapa banyak hasil yang ingin ditampilkan pada grafik. Pada sumbu x akan terdapat nama “*Author*”, sementara pada sumbu y akan terdapat jumlah kemunculan kata-kata dalam “`those_words`” pada *dataset*.

#### IV.5.2 Dashboard Cyber Threat Intelligence

*Dashboard Cyber Threat Intelligence* adalah sebuah tampilan antarmuka visual yang digunakan untuk memantau, menganalisis, dan memperoleh wawasan tentang ancaman siber atau *cyber threat*. *Dashboard* ini biasanya berisi informasi tentang keamanan sistem, jaringan, atau aplikasi yang diambil dari berbagai sumber data seperti *log*, *sensor*, dan *threat feed*. Tujuan dari *dashboard* ini adalah untuk

memberikan informasi yang cepat, akurat, dan terkini tentang keamanan informasi yang penting bagi organisasi. Dengan memanfaatkan teknologi visualisasi data, dashboard ini membantu para profesional keamanan informasi dalam memahami tren ancaman, pola serangan, dan kerentanan yang ada pada sistem mereka. Dengan demikian, organisasi dapat mengambil tindakan pencegahan yang tepat dan merespons ancaman siber secara lebih efektif.



Gambar IV-19 *Dashboard CTI*

Gambar IV-19 merupakan contoh visualisasi menggunakan *dashboard* yang mengumpulkan beberapa informasi grafis dan menyajukannya dalam satu lokasi.

## BAB V

### KESIMPULAN DAN SARAN

Bagian ini akan menjelaskan kesimpulan dan saran dari hasil pelaksanaan dan pengerjaan tugas akhir. Kesimpulan ditulis untuk menjelaskan dan menjawab beberapa pertanyaan pada rumusan masalah di Bab I. Saran ditulis untuk memberikan insight kepada penelitian dan pengembangan lebih lanjut.

#### V.1 Kesimpulan

Adapun kesimpulan dari rumusan masalah tugas akhir ini adalah.

1. Model *web scraper* yang dibangun dapat diterapkan untuk mengambil dan mengolah data yang dapat dimungkinkan untuk menyusun CTI proaktif. Akan tetapi, sumber data pada Reddit dibatasi pada subreddit tertentu yang dideklarasikan pada awal proses ekstraksi. Hal ini dikarenakan adanya keterbatasan pada API Reddit yang menerima masukan sebuah nama subreddit pada perintahnya. Meskipun terdapat keterbatasan pada sumber data yang dapat diakses melalui Reddit, penerapan model *web scraper* pada platform ini masih dapat memberikan manfaat dalam menyusun CTI proaktif. Dalam hal ini, *scraper* dapat digunakan untuk mengambil data dari subreddit - subreddit yang relevan dengan ancaman keamanan siber yang sedang dihadapi. Data-data tersebut kemudian dapat diolah dan dianalisis dengan menggunakan teknik-teknik *data science*, seperti *natural language processing* (NLP), untuk mengidentifikasi tren dan pola yang berkaitan dengan ancaman tersebut. Selain itu, penggunaan model *web scraper* pada Reddit juga dapat memberikan keuntungan dalam hal waktu dan biaya yang lebih efisien dibandingkan dengan metode pengumpulan data manual atau survei. Dengan demikian, model *web scraper* dapat menjadi alat yang berguna dalam memperoleh informasi CTI yang berkualitas dan memungkinkan organisasi untuk meningkatkan tingkat kesiapan mereka

dalam menghadapi ancaman keamanan siber yang semakin kompleks dan beragam.

2. Informasi yang dapat diperoleh dari proses *web scrapping* meliputi dua tabel *dataset*, yakni *dataset submission* dan *dataset* komentar. Kedua *dataset* tersebut dilakukan pemilahan field ataupun *parsing* sehingga menghasilkan tabel yang terstruktur. Maka dari itu, setiap tabel *dataset* mempunyai *field* atau atribut yang menjelaskan beberapa data yang relevan. *Dataset submission* umumnya terdiri dari *field* seperti judul submission, penulis, tanggal dibuat, jumlah *upvote* dan *downvote*, serta konten submission itu sendiri. Sedangkan pada *dataset* komentar, terdapat *field* seperti penulis komentar, jumlah *upvote* dan *downvote*, tanggal komentar, serta konten komentar itu sendiri. Dengan adanya informasi tersebut, dapat diolah menjadi bentuk visualisasi data yang dapat membantu dalam penyusunan *Cyber Threat Intelligence* (CTI) proaktif. Data-data tersebut dapat memberikan informasi terkait tren dan pola aktivitas, topik diskusi, serta sentimen pengguna Reddit terkait dengan suatu topik atau entitas tertentu.
3. Model web scraper dapat bekerja sesuai dengan analisis kebutuhan yang sudah dituliskan di Bab III. Berdasarkan hasil pembuatan model *web scraper* CTI, dapat disimpulkan bahwa model ini dapat bekerja secara efektif dan efisien sesuai dengan analisis kebutuhan yang sudah dituliskan di Bab III. Model ini mampu mengambil data dari subreddit tertentu dan menghasilkan *dataset submission* dan komentar yang terstruktur, yang kemudian dapat dimanfaatkan untuk memberikan informasi terkait tren keamanan siber.
4. Dalam pengolahan data dari model *web scraper* yang diterapkan pada Reddit, visualisasi data dapat menjadi langkah selanjutnya untuk mempermudah pemahaman terhadap data yang telah dihasilkan. Visualisasi tersebut dapat berupa grafik, diagram, *heatmap*, atau bentuk visualisasi lainnya yang dapat menunjukkan tren atau pola tertentu dari data yang

diperoleh. Namun, perlu diingat bahwa visualisasi data dapat diterapkan dengan lebih efektif apabila sumber data yang digunakan lebih luas dan *real-time* sehingga dapat memberikan gambaran yang lebih aktual dan akurat terhadap keadaan yang sedang terjadi.

## V.2 Saran

Berikut merupakan saran yang dapat digunakan sebagai panduan pengembangan lebih lanjut.

1. Pengolahan data pada CTI sangat penting untuk memberikan informasi yang akurat dan dapat diandalkan dalam memprediksi tren keamanan. Dengan menerapkan teknik *machine learning* pada tahap pengolahan data, akan memungkinkan organisasi untuk menghasilkan prediksi tren keamanan yang lebih presisi dan cepat dalam mengambil tindakan pencegahan terkait ancaman keamanan yang ada. Dalam hal ini, algoritma *machine learning* dapat digunakan untuk mengekstraksi pola dan tren dari data yang telah diolah, sehingga dapat memberikan hasil yang lebih akurat dan valid. Selain itu, dengan mengintegrasikan visualisasi pada proses pengolahan data, organisasi dapat memudahkan pemahaman dan pengambilan keputusan terkait tren keamanan yang muncul. Visualisasi yang baik dapat membantu analis keamanan untuk mengekstraksi informasi yang penting dan membuat keputusan yang cepat dan tepat. Oleh karena itu, implementasi *machine learning* dan visualisasi dalam CTI dapat menjadi faktor penting dalam menghasilkan prediksi keamanan yang efektif dan efisien bagi suatu organisasi.
2. Terdapat *gap knowledge* yang signifikan antara industri dan akademik dalam pemberian solusi keamanan siber melalui *Cyber Threat Intelligence* (CTI) di organisasi atau perusahaan. Hal ini disebabkan oleh perbedaan antara keadaan di lapangan yang lebih progresif dengan pengetahuan akademik yang cenderung konservatif. Salah satu solusinya adalah dengan membuat sebuah purwarupa sistem serupa yang dapat digunakan oleh

pelaku organisasi untuk menguji strategi keamanan siber sebelum diterapkan secara langsung. Selain itu, perlu dilakukan studi lebih lanjut mengenai kebutuhan keamanan siber di organisasi dan perusahaan agar penelitian dan pengembangan dapat dilakukan dengan lebih terfokus dan efektif. Dengan mengurangi *gap knowledge* antara industri dan akademik, diharapkan solusi CTI yang dihasilkan dapat lebih mewakili keadaan *real-time* dan efektif dalam mengatasi ancaman keamanan siber yang semakin kompleks dan dinamis.

## DAFTAR REFERENSI

- A framework for cyber threat hunting*. (2016). [online] Available: <https://www.threathunting.net/files/framework-for-threat-hunting-whitepaper.pdf>
- Almohannadi, H., Awan, I., Al Hamar, J., Cullen, A., Disso, J., & Armitage, L. (2018). *Cyber Threat Intelligence from Honeypot Data Using Elasticsearch*. 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA). pp. 900-906, doi: 10.1109/AINA.2018.00132.
- Amaro, L. J. B., Azevedo, B. W. P., de Mendonca, F. L. L., Giozza, W. F., Albuquerque, R. de, & Villalba, L. J. G. (2022). *Methodological framework to collect, process, analyze and visualize cyber threat intelligence data*. Applied Sciences, 12(3), 1205. <https://doi.org/10.3390/app12031205>.
- Ammari, T., Schoenebeck, S., & Romero, D. (2019). *Self-declared throwaway accounts on Reddit: How platform affordances and shared norms enable parenting disclosure and support*. Proceedings of the ACM on Human-Computer Interaction, 3(CSCW), 1–30.
- Chandrasekharan, E., Samory, M., Jhaver, S., Charvat, H., Bruckman, A., Lampe, C., Eisenstein, J., & Gilbert, E. (2018). *The Internet's hidden rules: An empirical study of Reddit norm violations at micro, meso, and macro scales*. Proceedings of the ACM on Human-Computer Interaction, 2, 32.
- Chismon, D., & Ruks, M. (2015). *Threat intelligence: Collecting, analysing, evaluating*. MWR InfoSecurity Ltd, 3(2), 36-42
- Ciobanu, C., Dandurand, L., Grobauer, M., Kacha, B., Kaplan, P., Kompanek, A., & Van Horenbeeck, M. (2014). *Actionable Information for Security Incident Response*. ENISA, Heraklion, Greece
- Dusch, B. (2022). *UI and UX Design: Design thinking*. Codecademy. <https://www.codecademy.com/resources/docs/uiux/design-thinking>
- Farnham, G., & Leune, K. (2013). *Tools and standards for cyber threat intelligence projects*. SANS Institute, 3(2), 25-31
- Feng, B. (2021, August 20). *Threat intelligence sharing: What kind of intelligence to share?* Retrieved October 5, 2022, from <https://www.concordia-h2020.eu/blog-post/threat-intelligence-sharing/>
- Ferrara, E., De Meo, P., Fiumara, G., & Baumgartner, R. (2014). *Web data extraction, applications and techniques: A survey*. Knowledge-Based Systems. 70. 301–323. <https://doi.org/10.1016/j.knosys.2014.07.007>
- Fiesler, C., Beard, N., & Keegan, B. C. (2020). *No robots, spiders, or scrapers: Legal and ethical regulation of data collection methods in social media*

- terms of service*. Proceedings of the International AAAI Conference on Web and Social Media, 14, 187–196.
- Fu, T., Abbasi, A., & Chen, H. (2010). *A Focused Crawler for Dark Web Forums*. Journal of the American Society for Information Science and Technology. [online] Available: <https://doi.org/10.1002/asi>
- Hilbert, M. (2015). *Big data for development: A review of promises and challenges*. Development Policy Review. pp. 10-07.
- Jiang, J., Song, X., Yu, N., & Lin, C. Y. (2014). *FoCUS?: Learning to Crawl Web Forums*. IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 6, pp. 1293-1306
- Johnson, C., Badger, L., Waltermire, D., Snyder, J., & Skorupka, C. (2016, October). *Guide to cyber threat information sharing - NIST*. National Institute of Standards and Technology. Retrieved October 1, 2022, from <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-150.pdf>
- Lee, R., Assante, M., & Conway, T. (2016, March). *Analysis of the Cyber Attack on the Ukrainian Power Grid*. Electricity Information Sharing and Analysis Center. Retrieved October 5, 2022, from [https://paper.seebug.org/papers/APT/APT\\_CyberCriminal\\_Campagin/2016/2016.03.18.Analysis\\_of\\_the\\_Cyber\\_Attack\\_on\\_the\\_Ukrainian\\_Power\\_Grid/E-ISAC\\_SANS\\_Ukraine\\_DUC\\_5.pdf](https://paper.seebug.org/papers/APT/APT_CyberCriminal_Campagin/2016/2016.03.18.Analysis_of_the_Cyber_Attack_on_the_Ukrainian_Power_Grid/E-ISAC_SANS_Ukraine_DUC_5.pdf)
- Marres, N., & Weltevrede, E. (2013). *Scraping the social?* Journal of Cultural Economy. 6(3). 313–335. <https://doi.org/10.1080/17530350.2013.772070>
- Medvedev, A. N., Lambiotte, R., & Delvenne, J.-C. (2019). *The Anatomy of Reddit: An Overview of Academic Research*. Springer Proceedings in Complexity, 183–204. doi:10.1007/978-3-030-14683-2\_9
- Nunes, E. (2016). *Darknet and deepnet mining for proactive cybersecurity threat intelligence*. 2016 IEEE Conference on Intelligence and Security Informatics (ISI). pp. 7-12, doi: 10.1109/ISI.2016.7745435.
- Parvez, M. S., Tasneem, K. S., Rajendra, S. S., & Bodke, K. R. (2018). *Analysis of different web data extraction techniques*. 2018 International Conference on Smart City and Emerging Technology (ICSCET). <https://doi.org/10.1109/icscet.2018.8537333>
- Pavkovic, M. & Protic, J. (2013). *Intelligent crawler for web forums based on improved regular expressions*. 2013 21st Telecommunications Forum Telfor TELFOR 2013 - Proceedings of Papers, pp. 817-820, [online] Available: <https://doi.org/10.1109/TELFOR.2013.6716355>.
- Portokalidis, G., Slowinska, A., & Bos, H. (2006). *Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic*



- signature generation*. ACM SIGOPS Operating Systems Review, ACM. vol. 40. pp. 15-27
- Proferes, N., Jones, N., Gilbert, S., Fiesler, C., & Zimmer, M. (2021). *Studying Reddit: A Systematic Overview of Disciplines, Approaches, Methods, and Ethics*. Social Media + Society, 7(2). <https://doi.org/10.1177/20563051211019004>
- Reddit.com. (2020). *User agreement—October 15, 2020—Reddit*. Reddit User Agreement. <https://www.redditinc.com/policies/user-agreement-october-15-2020>
- Reddit.com. (2021, January 17). *Advertising—Audience—Reddit*. Discover what makes Reddit ads unique. <https://web.archive.org/web/20210117184818/https://www.redditinc.com/advertising/audience>
- Reddit.com. (2023). *API Documentation*. Reddit.com: API documentation. <https://www.reddit.com/dev/api/>
- Samtani, S., Chinn, R., Chen, H., & Nunamaker, J. F. (2017). *Exploring Emerging Hacker Assets and Key Hackers for Proactive Cyber Threat Intelligence*. Journal of Management Information Systems, 34(4), 1023–1053. doi:10.1080/07421222.2017.1394049
- Shackleford, D. (2015). *Who's using cyberthreat intelligence and how?* SANS Institute. From [www.sans.org/reading-room/whitepapers/analyst/cyberthreat-intelligence-how35767](http://www.sans.org/reading-room/whitepapers/analyst/cyberthreat-intelligence-how35767)
- Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., & Nakao, K. (2011). *Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation*. Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security pp. 29-36
- Wagner, T. D., Mahbub, K., Palomar, E., & Abdallah, A. E. (2019). *Cyber threat intelligence sharing: Survey and research directions*. Computers & Security, 87. <https://doi.org/10.1016/j.cose.2019.101589>
- Williams, R., Samtani, S., Patton, M., and Chen, H. (2018). *Incremental Hacker Forum Exploit Collection and Classification for Proactive Cyber Threat Intelligence: An Exploratory Study*. 2018 IEEE International Conference on Intelligence and Security Informatics (ISI). pp. 94-99. doi: 10.1109/ISI.2018.8587336

## Lampiran A. Source Code Program

### A.1 subreddit\_downloader.py

```
import sys
import csv
import json
import praw
import yaml
import typer
from datetime import datetime

# noinspection PyUnresolvedReferences
import pretty_errors # keep the import to have better
error messages

from os.path import join
from pathlib import Path
from typer import Argument
from typer import Option
from typing import Optional, List
from loguru import logger
from codetiming import Timer
from pushshift_py import PushshiftAPI
from prawcore.exceptions import NotFound

class OutputManager:
    """
    Class used to collect and store data (submissions
```

```

and comments)
    """
    params_filename = "params.yaml"

    def __init__(self, output_dir: str, subreddit:
str):
        self.submissions_list = []
        self.submissions_raw_list = []
        self.comments_list = []
        self.comments_raw_list = []
        self.run_id =
datetime.today().strftime('%Y%m%d%H%M%S')

        self.subreddit_dir = join(output_dir,
subreddit)
        self.runtime_dir = join(self.subreddit_dir,
self.run_id)

        self.submissions_output =
join(self.runtime_dir, "submissions")
        self.sub_raw_output = join(self.runtime_dir,
"submissions", "raw")
        self.comments_output = join(self.runtime_dir,
"comments")
        self.comments_raw_output =
join(self.runtime_dir, "comments", "raw")
        self.params_path = join(self.runtime_dir,
OutputManager.params_filename)

        self.total_submissions_counter = 0

```

```

        self.total_comments_counter = 0

        for path in [self.submissions_output,
                     self.sub_raw_output,
                     self.comments_output,
                     self.comments_raw_output]:
            Path(path).mkdir(parents=True,
exist_ok=True)

    def reset_lists(self):
        self.submissions_list = []
        self.submissions_raw_list = []
        self.comments_list = []
        self.comments_raw_list = []

    def store(self, lap: str):
        # Track total data statistics
        self.total_submissions_counter +=
len(self.submissions_list)
        self.total_comments_counter +=
len(self.comments_list)

        # Store the collected data
        dictlist_to_csv(join(self.submissions_output,
f"{lap}.csv"), self.submissions_list)
        dictlist_to_csv(join(self.comments_output,
f"{lap}.csv"), self.comments_list)

        if len(self.submissions_raw_list) > 0:
            with open(join(self.sub_raw_output,

```

```

f"{lap}.njson"), "a", encoding="utf-8") as f:
    f.write("\n".join(json.dumps(row) for
row in self.submissions_raw_list))
    if len(self.comments_raw_list) > 0:
        with open(join(self.comments_raw_output,
f"{lap}.njson"), "a", encoding="utf-8") as f:
            f.write("\n".join(json.dumps(row,
default=lambda o: '<not serializable>')
                                for row in
self.comments_raw_list))

    def store_params(self, params: dict):
        with open(self.params_path, "w", encoding="utf-
8") as f:
            yaml.dump(params, f)

    def load_params(self) -> dict:
        with open(self.params_path, "r", encoding="utf-
8") as f:
            params = yaml.load(f, yaml.FullLoader)
            return params

    def enrich_and_store_params(self, utc_older: int,
utc_newer: int):
        params = self.load_params()
        params["utc_older"] = utc_older
        params["utc_newer"] = utc_newer
        params["total_comments_counter"] =
self.total_comments_counter
        params["total_submissions_counter"] =

```

```

self.total_submissions_counter
    params["total_counter"] =
self.total_comments_counter +
self.total_submissions_counter
    self.store_params(params)

def dictlist_to_csv(file_path: str, dictionaries_list:
List[dict]):
    if len(dictionaries_list) == 0:
        dictionaries_list = [{}]
    keys = dictionaries_list[0].keys()
    with open(file_path, 'w', newline='',
encoding="utf-8") as output_file:
        dict_writer = csv.DictWriter(output_file, keys,
dialect="excel")
        dict_writer.writeheader()
        dict_writer.writerows(dictionaries_list)

def init_locals(debug: str,
                output_dir: str,
                subreddit: str,
                utc_upper_bound: str,
                utc_lower_bound: str,
                run_args: dict,
                ) -> (str, OutputManager):
    assert not (utc_upper_bound and utc_lower_bound),
    "`utc_lower_bound` and " \

```

```

    "`utc_upper_bound` parameters are in mutual exclusion"
    run_args.pop("reddit_secret")

    if not debug:
        logger.remove()
        logger.add(sys.stderr, level="INFO")

    direction = "after" if utc_upper_bound else
"before"
    output_manager = OutputManager(output_dir,
subreddit)

    output_manager.store_params(run_args)
    return direction, output_manager

def init_clients(reddit_id: str,
                 reddit_secret: str,
                 reddit_username: str
                 ) -> (PushshiftAPI, praw.Reddit):
    pushshift_api = PushshiftAPI()

    reddit_api = praw.Reddit(
        client_id=reddit_id,
        client_secret=reddit_secret,

user_agent=f"python_script:subreddit_downloader:(by
/u/{reddit_username})",
    )

```

```

    return pushshift_api, reddit_api

def utc_range_calculator(utc_received: int,
                          utc_upper_bound: int,
                          utc_lower_bound: int
                          ) -> (int, int):
    """
    Calculate the max UTC range seen.

    Increase/decrease utc_upper_bound/utc_lower_bound
    according with utc_received value
    """
    if not utc_upper_bound or not utc_lower_bound:
        utc_upper_bound = utc_received
        utc_lower_bound = utc_received

    utc_lower_bound = utc_lower_bound if utc_received >
    utc_lower_bound else utc_received
    utc_upper_bound = utc_upper_bound if utc_received <
    utc_upper_bound else utc_received

    return utc_lower_bound, utc_upper_bound

def comments_fetcher(sub, output_manager, reddit_api,
                     comments_cap):
    """
    Comments fetcher

    Get all comments with depth-first approach

```



```

        Solution from
https://praw.readthedocs.io/en/latest/tutorials/comments.html

        """

        try:
            submission_rich_data =
reddit_api.submission(id=sub.id)
            logger.debug(f"Requesting
{submission_rich_data.num_comments} comments...")

            submission_rich_data.comments.replace_more(limit=commen
ts_cap)

            comments = submission_rich_data.comments.list()
        except NotFound:
            logger.warning(f"Submission not found in PRAW:
`{sub.id}` - `{sub.title}` - `{sub.full_link}`")

            return

        for comment in comments:
            comment_useful_data = {
                "id": comment.id,
                "submission_id": sub.id,
                "created_utc": int(comment.created_utc),
                "Author": comment.Author,
                "score": comment.score,
                "body": comment.body.replace('\n', '\\n'),
                "parent_id": comment.parent_id,
                "permalink": comment.permalink,
            }

            output_manager.comments_raw_list.append(comment.__dict__

```

```

_)

output_manager.comments_list.append(comment_useful_data
)

def submission_fetcher(sub, output_manager:
OutputManager):
    """
    Get and store reddit submission info
    """
    # Sometimes the submission doesn't have the
selftext
    self_text_normalized = sub.selftext.replace('\n',
'\n') if hasattr(sub, "selftext") else "<not selftext
available>"

    submission_useful_data = {
        "id": sub.id,
        "created_utc": int(sub.created_utc),
        "Author": sub.Author,
        "num_comments": sub.num_comments,
        "title": sub.title.replace('\n', '\n'),
        "selftext": self_text_normalized,
        "full_link": sub.full_link,
    }

    output_manager.submissions_list.append(submission_usefu
l_data)
    output_manager.submissions_raw_list.append(sub.d_)

```

```

class HelpMessages:
    help_reddit_url = "https://github.com/reddit-
archive/reddit/wiki/OAuth2"
    help_reddit_agent_url = "https://github.com/reddit-
archive/reddit/wiki/API"
    help_praw_replace_more_url =
"https://asynpraw.readthedocs.io/en/latest/code_overvi
ew/other/commentforest.html#asynpraw.models.comment_fo
rest.CommentForest.replace_more"

    subreddit = "The subreddit name"
    output_dir = "Optional output directory"
    batch_size = "Request `batch_size` submission per
time"
    laps = "How many times request `batch_size` reddit
submissions"
    reddit_id = f"Reddit client_id, visit
{help_reddit_url}"
    reddit_secret = f"Reddit client_secret, visit
{help_reddit_url}"
    reddit_username = f"Reddit username, used for build
the `user_agent` string, visit {help_reddit_agent_url}"
    utc_after = "Fetch the submissions after this UTC
date"
    utc_before = "Fetch the submissions before this UTC
date"
    debug = "Enable debug logging"
    comments_cap = f"Some submissions have 10k> nested

```

```

comments and stuck the praw API call." \
    f"If provided, the system requires
new comments `comments_cap` times to the praw API." \
    f"`comments_cap` under the hood will
be passed directly to `replace_more` function as " \
    f"`limit` parameter. For more info
see the README and visit {help_praw_replace_more_url}."

# noinspection PyTypeChecker
@Timer(name="main", text="Total downloading time:
{minutes:.1f}m", logger=logger.info)
def main(subreddit: str = Argument(...,
help=HelpMessages.subreddit),
        output_dir: str = Option("./data/",
help=HelpMessages.output_dir),
        batch_size: int = Option(10,
help=HelpMessages.batch_size),
        laps: int = Option(3, help=HelpMessages.laps),
        reddit_id: str = Option(...,
help=HelpMessages.reddit_id),
        reddit_secret: str = Option(...,
help=HelpMessages.reddit_secret),
        reddit_username: str = Option(...,
help=HelpMessages.reddit_username),
        utc_after: Optional[str] = Option(None,
help=HelpMessages.utc_before),
        utc_before: Optional[str] = Option(None,
help=HelpMessages.utc_before),
        comments_cap: Optional[int] = Option(None,

```

```

help=HelpMessages.comments_cap),
    debug: bool = Option(False,
help=HelpMessages.debug),
):
    """
    Download all the submissions and relative comments
    from a subreddit.
    """

    # Init
    utc_upper_bound = utc_after
    utc_lower_bound = utc_before
    direction, out_manager = init_locals(debug,
                                         output_dir,
                                         subreddit,

    utc_upper_bound,

    utc_lower_bound,

    run_args=locals())
    pushshift_api, reddit_api = init_clients(reddit_id,
    reddit_secret, reddit_username)
    logger.info(f"Start download: "
                f"UTC range: [{utc_lower_bound},
    {utc_upper_bound}], "
                f"direction: `{direction}`, "
                f"batch size: {batch_size}, "
                f"total submissions to fetch:
    {batch_size * laps}")

```

```

# Start the gathering
for lap in range(laps):
    logger.debug(f"New lap start: {lap}")
    lap_message = f"Lap {lap}/{laps} completed in
    \"{minutes:.1f}m | " \
        f"[new/tot]:
    {len(out_manager.comments_list)}/{out_manager.total_comments_counter}"

    with Timer(text=lap_message,
logger=logger.info):
        # Reset the data already stored
        out_manager.reset_lists()

        # Fetch data in the `direction` way
        submissions_generator =
pushshift_api.search_submissions(subreddit=subreddit,

limit=batch_size,

sort='desc' if direction == "before" else 'asc',

sort_type='created_utc',

after=utc_upper_bound if direction == "after" else
None,

before=utc_lower_bound if direction == "before" else
None,

```

```

)

    for sub in submissions_generator:
        logger.debug(f"New submission
`{sub.full_link}` - created_utc: {sub.created_utc}")

        # Fetch the submission data
        submission_fetcher(sub, out_manager)

        # Fetch the submission's comments
        comments_fetcher(sub, out_manager,
reddit_api, comments_cap)

        # Calculate the UTC seen range
        utc_lower_bound, utc_upper_bound =
utc_range_calculator(sub.created_utc,

utc_upper_bound,

utc_lower_bound)
        # Store data (submission and comments)
        out_manager.store(lap)

        # Check the bounds
        assert utc_lower_bound < utc_upper_bound,
f"utc_lower_bound '{utc_lower_bound}' should be " \

f"less than utc_upper_bound '{utc_upper_bound}'"
        logger.debug(f"utc_upper_bound:

```

```
{utc_upper_bound} , utc_lower_bound:
{utc_lower_bound}")

out_manager.enrich_and_store_params(utc_newer=utc_upper
_bound, utc_older=utc_lower_bound)
    logger.info(f"Stop download: lap {laps}/{laps}
[total]: {out_manager.total_comments_counter}")

if __name__ == '__main__':
    typer.run(main)
```



## A.2 dataset\_builder.py

```
import sys
import csv
import typer
import pretty_errors  # keep the import to have better
error messages

from os import listdir
from os.path import isfile
from os.path import join
from loguru import logger
from typer import Option
from datetime import datetime
from pathlib import Path
from typing import List
from typing import Set
from typing import Optional
from typing import Tuple
from tqdm import tqdm

class DatasetManager:
    subreddit_header_name = "subreddit"

    def __init__(self, output_path: str, caching_size:
int):
        self.comments_rows = []
        self.submissions_rows = []
        self.total_comments = 0
        self.total_submissions = 0
```

```

        self.comments_census_ids = set()
        self.submissions_census_ids = set()
        self.subreddit_name = "undefined"
        self.output_path = output_path
        self.caching_size = caching_size

        self.comments_csv_header: Optional[List[str]] =
None
        self.submissions_csv_header:
Optional[List[str]] = None

        self.run_id =
datetime.today().strftime('%Y%m%d%H%M%S')
        self.runtime_dir = join(self.output_path,
self.run_id)
        self.comments_output_path =
join(self.runtime_dir, "comments.csv")
        self.submissions_output_path =
join(self.runtime_dir, "submissions.csv")
        Path(self.runtime_dir).mkdir(parents=True,
exist_ok=True)

    def set_subreddit(self, subreddit_name: str):
        self.subreddit_name = subreddit_name

    def set_comments_csv_header(self, comments_header:
List[str]):
        if not self.comments_csv_header:
            self.comments_csv_header =
[DatasetManager.subreddit_header_name] +

```

```

comments_header

    def set_submissions_csv_header(self,
comments_header: List[str]):
        if not self.submissions_csv_header:
            self.submissions_csv_header =
[DatasetManager.subreddit_header_name] +
comments_header

    def _flush_comments(self):
        self.store_comments()
        self.comments_rows = []

    def _flush_submissions(self):
        self.store_submissions()
        self.submissions_rows = []

    def _rows_parser(self,
                        rows: List[List[str]],
                        census_ids: Set[str],
                        header: List[str]):
        # TODO find a more efficient way to store only
unique `id`,
        # maybe leveraging the Python set structure
        rows_to_remove = []
        for idx, row in enumerate(rows):
            if not row: # see README.md:notes:
                continue
            row.insert(0, self.subreddit_name)
            row_values = dict(zip(header, row))

```

```

        row_id = row_values["id"]
        if row_id in census_ids:
            logger.debug(f"Find duplicate id
'{row_id}'")

            rows_to_remove.append(idx)
            census_ids.add(row_id)

        for idx in sorted(rows_to_remove,
reverse=True): # https://stackoverflow.com/a/11303234
            del rows[idx]

    def populate_comments(self, rows: List[List[str]]):
        self._rows_parser(rows,
self.comments_census_ids, self.comments_csv_header)
        self.total_comments += len(rows)
        self.comments_rows.extend(rows)
        if len(self.comments_rows) > self.caching_size:
            self._flush_comments()

    def populate_submissions(self, rows:
List[List[str]]):
        self._rows_parser(rows,
self.submissions_census_ids,
self.submissions_csv_header)
        self.total_submissions += len(rows)
        self.submissions_rows.extend(rows)
        if len(self.submissions_rows) >
self.caching_size:
            self._flush_submissions()

```

```

    def store_comments(self):
        csv_writer(self.comments_output_path,
self.comments_csv_header, self.comments_rows)

    def store_submissions(self):
        csv_writer(self.submissions_output_path,
self.submissions_csv_header, self.submissions_rows)

def init(debug: bool):
    logger.add(lambda msg: tqdm.write(msg, end=""))
    if not debug:
        logger.remove()
        logger.add(sys.stderr, level="INFO")

def csv_writer(csv_path: str, header: List[str], rows:
List[List[str]]):
    skip_header = isfile(csv_path) # If file already
exist, don't write the header

    with open(csv_path, "a+", newline='',
encoding="utf-8") as f:
        logger.debug(f"Storing data on '{csv_path}' -
header: '{header}'")
        file_writer = csv.writer(f, dialect="excel")
        if not skip_header:
            file_writer.writerow(header)
        file_writer.writerows(rows)

```

```

def csv_reader(csv_path: str) -> Tuple[List[str],
List[List[str]]]:
    header = None
    rows = []
    with open(csv_path, newline='', encoding="utf-8")
as csv_file:
        file_reader = csv.reader(csv_file,
dialect="excel")
        for row_id, row in enumerate(file_reader):
            if row_id == 0:
                header = row
                logger.debug(f"File '{csv_path}'
header: '{header}'")
                continue
            rows.append(row)
    return header, rows

class HelpMessages:
    """
    Utility class to try maintain clean the `main`
function signature
    """
    input_dir = "Main directory with scraped data and
this structure: " \
        "`/<subreddit>/<timestamp>/[comments |
submission]`"
    output_dir = "Optional output directory"
    config_size = "Store data on the output each

```

```

`caching_size` number of comments"
    debug = "Enable debug logging"

def main(input_dir: str = Option("./data/",
help=HelpMessages.input_dir),
        output_path: str = Option("./dataset/",
help=HelpMessages.output_dir),
        caching_size: int = Option(1000,
help=HelpMessages.config_size),
        debug: bool = Option(False,
help=HelpMessages.debug),
        ):
    init(debug)
    dataset_mng = DatasetManager(output_path,
caching_size)

    for subreddit_name in tqdm(listdir(input_dir)): #
<input_dir>/<sub>
        subreddit_path = join(input_dir,
subreddit_name)
        if isfile(subreddit_path):
            continue
        logger.debug(f"Start parsing subreddit
`{subreddit_name}` data")
        dataset_mng.set_subreddit(subreddit_name)

        for job_id in tqdm(listdir(subreddit_path)): #
<input_dir>/<sub>/<job>
            job_folder_path = join(subreddit_path,

```

```

job_id)

        comments_folder_path =
join(job_folder_path, "comments")
        submissions_folder_path =
join(job_folder_path, "submissions")

        for csv_filename in
listdir(comments_folder_path): #
<input_dir>/<sub>/<job>/comments/<file>.csv
            csv_path = join(comments_folder_path,
csv_filename)

            if not isfile(csv_path):
                continue

            header, rows = csv_reader(csv_path)

dataset_mng.set_comments_csv_header(header)
        dataset_mng.populate_comments(rows)
        logger.debug(f"Comments for job `{job_id}`
loaded")

        for csv_filename in
listdir(submissions_folder_path): #
<input_dir>/<sub>/<job>/submissions/<file>.csv
            csv_path =
join(submissions_folder_path, csv_filename)

            if not isfile(csv_path):
                continue # skip `raw` folder
            header, rows = csv_reader(csv_path)

dataset_mng.set_submissions_csv_header(header)

```



```
        dataset_mng.populate_submissions(rows)
        logger.debug(f"Submissions for job
`{job_id}` loaded")

        logger.debug(f"Storing data for
`{subreddit_name}`")
        dataset_mng.store_comments()
        dataset_mng.store_submissions()

if __name__ == '__main__':
    typer.run(main)
```

### A.3 dataprocess.ipynb

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

import re
import string
import nltk
```

```
Membuat dataframe, dan menghapus salah satu kolom yang
dobel
df = pd.read_csv("../rawdata/hololivecomments.csv")
del df['subreddit.1']
```

```
# Cek tipe data tabel dan kolom-kolomnya serta salah
satu sampel
df.info()
df.sample(1)
```

```
# Cek baris yang terduplikasi
print("\nJumlah baris yang terduplikasi:",
df.duplicated().sum())
```

```
# Hapus baris yang terduplikasi
df.drop_duplicates(inplace=True)
```

```
# Ada kondisi dimana post telah dihapus, sehingga
kolom body menjadi [removed]. Ubah data tersebut
menjadi nilai NA supaya mudah dianalisis
df['body'] = df['body'].replace('[removed]', np.nan)
```

```
# Cek berapa nilai NA di setiap tabel
df.isna().sum()
```

```
# Lihat sampel data yang terdapat nilai NA
nan_row = df[df.isna().any(axis=1)].sample(1)

nan_row
```

```
# Ganti nilai NA tersebut dengan string NaN, asumsi:
username NaN di Reddit sudah tidak aktif dan tidak
memberikan post ke subreddit yang dianalisis
df['Author'] = df['Author'].fillna('NaN')
```

```
# Cek berapa nilai NA di setiap tabel
df.isna().sum()
```

```
# Hapus semua baris yang ada nilai NA dari tabel
df.dropna(axis=0, how="any", subset=None,
inplace=True)
```

```
# Cek berapa nilai NA di setiap tabel
df.isna().sum()
```

```
# Sort data berdasarkan waktu pos dan mengubah epoch  
value menjadi datetime UTC+7  
  
df.sort_values(by = ["created_utc"], inplace = True,  
ascending = True)  
df['created_utc'] =  
pd.DatetimeIndex(pd.to_datetime(df['created_utc'],  
unit='s')).tz_localize('UTC').tz_convert('Asia/Jakarta')
```

```
df.info()
```

```
# Membuat matriks korelasi  
corr = df.corr()  
plt.figure(figsize=(12,10))  
sns.heatmap(corr, annot=True, cmap=plt.cm.Reds)  
plt.title("Correlation Matrix", fontsize=16)  
plt.show()
```

```
nltk.download('punkt')
nltk.download('stopwords')

STOP_WORDS = stopwords.words()

# removing the emojis
# https://www.kaggle.com/alankritamishra/covid-19-
# tweet-sentiment-analysis#Sentiment-analysis
EMOJI_PATTERN = re.compile("[  

                                u"\U0001F600-\U0001F64F"  

# emoticons  

                                u"\U0001F300-\U0001F5FF"  

# symbols & pictographs  

                                u"\U0001F680-\U0001F6FF"  

# transport & map symbols  

                                u"\U0001F1E0-\U0001F1FF"  

# flags (iOS)
```

```

u"\U00002702-\U000027B0"
u"\U000024C2-\U0001F251"
"]+", flags=re.UNICODE)

def cleaning(text):
    """
    Convert to lowercase.
    Remove URL links, special characters and
    punctuation.
    Tokenize and remove stop words.
    """
    text = text.lower()
    text = re.sub('https?:\/\/\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('[\'\""...]', '', text)

    text = EMOJI_PATTERN.sub(r'', text)

    # removing the stop-words

    text_tokens = word_tokenize(text)

    tokens_without_sw = [word for word in text_tokens
if not word in STOP_WORDS]

    filtered_sentence = (" ").join(tokens_without_sw)

    text = filtered_sentence

    return text

```

```
# Lakukan fungsi cleaning di dataframe dan dump ke CSV
df2 = df.copy()

df2['body'] = df2['body'].apply(cleaning)
df2.to_csv("../rawdata/hololivecomments_cleaned1.csv", index=False)
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Define the x and y data
x = np.array([0.1, 18.3, 109.4, 175.6, 340.6, 722.8, 1075.2, 1407.4, 1824.7, 3310.8])
y = np.array([157, 5957, 40486, 54697, 117885, 236340, 345979, 421088, 592810, 1035939])

# Reshape the x data to fit the format expected by scikit-learn
x = x.reshape(-1, 1)

# Create and fit the linear regression model
model = LinearRegression().fit(x, y)
```

```

# Print the slope and intercept of the regression
line
slope = model.coef_[0]
intercept = model.intercept_
print(f"Regression Formula: y = {slope:.4f}x +
{intercept:.4f}")

# Print the slope and intercept of the regression
line
slope = model.coef_[0]
intercept = model.intercept_
print(f"Regression Formula: y = {slope:.4f}x +
{intercept:.4f}")

# Create a scatter plot of the data
plt.scatter(x, y)

# Add the regression line to the plot
plt.plot(x, model.predict(x), color='red')

# Add labels and legend to the plot
plt.xlabel('Waktu Jalannya Program (menit)')
plt.ylabel('Jumlah Item yang diperoleh (juta)')
plt.title('Regresi Linier Keterhubungan Jumlah Item
Berdasarkan Lama Waktu')

# Show the plot
plt.show()

```



```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Define the first set of x and y data
x1 = np.array([5956, 32128, 58256, 86960, 117877,
146100, 177287, 207165, 236340])
y1 = np.array([15.4, 83.4, 150.2, 223.8, 303.6,
376.6, 456.8, 533.9, 609.7])

# Define the second set of x and y data
x2 = np.array([5956, 32128, 58256, 86960, 117877,
146100, 177287, 207165, 236340])
y2 = np.array([13.9, 75.2, 135.8, 202.3, 274.4,
340.4, 413, 482.7, 551.2])

# Define the third set of x and y data
x3 = np.array([5956, 32128, 58256, 86960, 117877,
146100, 177287, 207165, 236340])
y3 = np.array([1.5, 8.2, 14.4, 21.5, 29.2, 36.2,
43.8, 51.2, 58.5])

# Reshape the x data to fit the format expected by
scikit-learn
x1 = x1.reshape(-1, 1)
x2 = x2.reshape(-1, 1)
x3 = x3.reshape(-1, 1)

```

```
# Create and fit the linear regression model for the
first set of data
model1 = LinearRegression().fit(x1, y1)

# Print the slope and intercept of the regression
line for the first set of data
slope1 = model1.coef_[0]
intercept1 = model1.intercept_
print(f"Regression Formula (Total): y = {slope1:.4f}x
+ {intercept1:.4f}")

# Create and fit the linear regression model for the
second set of data
model2 = LinearRegression().fit(x2, y2)

# Print the slope and intercept of the regression
line for the second set of data
slope2 = model2.coef_[0]
intercept2 = model2.intercept_
print(f"Regression Formula (JSON): y = {slope2:.4f}x
+ {intercept2:.4f}")

# Create and fit the linear regression model for the
third set of data
model3 = LinearRegression().fit(x3, y3)
```

```

# Print the slope and intercept of the regression
line for the third set of data
slope3 = model3.coef_[0]
intercept3 = model3.intercept_
print(f"Regression Formula (CSV): y = {slope3:.4f}x +
{intercept3:.4f}")

# Create a scatter plot of both datasets
plt.scatter(x1, y1, color="red", label="Total")
plt.scatter(x2, y2, color="green", label="JSON")
plt.scatter(x3, y3, color="blue", label="CSV")

# Add the regression lines to the plot
plt.plot(x1, model1.predict(x1), color="red",
label="Total")
plt.plot(x2, model2.predict(x2), color="green",
label="JSON")
plt.plot(x3, model2.predict(x3), color="blue",
label="CSV")

# Add labels and legend to the plot
plt.xlabel('Jumlah Unit yang diterima')
plt.ylabel('Size (MB) ')
plt.title('Regresi Linier Keterhubungan Ukuran Data
Berdasarkan Jumlah Unit')
plt.legend()

# Show the plot
plt.show()

```

```

import matplotlib.pyplot as plt
import numpy as np

# Define the x values to plot
x = np.linspace(0, 5000, 1000)

# Define the equation of the line
y = 0.81328*x

# Create a figure and axis object
fig, ax = plt.subplots()

# Plot the line
ax.plot(x, y)

# Set the axis labels and title
ax.set_xlabel('Waktu Jalannya Program (menit)')
ax.set_ylabel('Size (MB)')
# ax.set_title('Line Plot of y = 0.81328x + 4108.7')

# Show the plot
plt.show()

```

#### A.4 dataviz.ipynb

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import re
import string
import nltk

from collections import Counter
from nltk import tokenize
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

```
dv =
pd.read_csv("../rawdata/hololivecomments_cleaned1.csv")
```

```
dv["created_utc"] =
pd.to_datetime(dv["created_utc"]).dt.tz_convert(None)
```

```
# Check dataframe

display(dv["submission_id"].count())
display(dv.info())
display(dv.head(1))
```

```

# Function and Procedure

tokenSpace = tokenize.WhitespaceTokenizer()

def counter(text, columnText, quantity): #
    counter(dv[dv["score"] == 1], "body", 10)
    allWords = ' '.join([text for text in
text[columnText].astype('str')])
    tokenPhrase = tokenSpace.tokenize(allWords)
    frequency = nltk.FreqDist(tokenPhrase)
    dfFrequency = pd.dataframe({columnText:
list(frequency.keys()), "Frequency":
list(frequency.values())})

    dfFrequency = dfFrequency.nlargest(columns =
"Frequency", n = quantity)
    plt.figure(figsize=(12,8))
    ax = sns.barplot(data = dfFrequency, x =
columnText, y = "Frequency", palette="deep")
    ax.set(ylabel = "Count")
    ax.set_title('Top {} in {} Where (Certain
Condition)'.format(quantity, columnText))
    plt.xticks(rotation='horizontal')
    plt.show()

```

```

def wordappear_permonth(word):
    # Make a data copy, so the origin of data won't
    changed
    dv1 = dv.copy()

    # convert the body column to strings to ensure
    that the regular expression search works correctly
    dv1['body'] = dv1['body'].astype(str)

    # create a new column that contains a boolean
    indicating whether the word appears in the body
    attribute
    dv1['contains_word'] = dv1['body'].apply(lambda
x: True if re.search(r'\b{}\b'.format(word), x,
re.IGNORECASE) else False)

    # group the dataframe by the year and month of
    the created_utc attribute, and count the number of
    rows in each group where the word appears in the body
    word_counts = dv1[dv1['contains_word'] ==
True].groupby(dv1['created_utc'].dt.to_period('M')).s
ize()

    # convert the Period objects to timestamps
    word_counts.index =
word_counts.index.to_timestamp()

```

```
# create the bar plot
fig, ax = plt.subplots(figsize=(12,6))
ax.bar(word_counts.index, word_counts, width=20)
ax.set_xlabel('Month')
ax.set_ylabel('Number of Appearances of
("{}").format(word))
ax.set_title('Number of "{}" Appearances by
Month'.format(word))

# rotate x-axis labels
plt.xticks(rotation=90, ha='right')

plt.show()
```



```

def numberofcomments_permonth():
    # Make a data copy, so the origin of data won't
    changed
    dv1 = dv.copy()

    # Set created_utc column as the index
    dv1.set_index('created_utc', inplace=True)

    # Group the rows by month and count the number of
    rows per month
    monthly_counts = dv1.resample('M').size()

    # Plot the graph
    plt.subplots(figsize=(12,6))
    plt.plot(monthly_counts.index,
monthly_counts.values)
    plt.title('Number of Rows per Month')
    plt.xlabel('Month')
    plt.ylabel('Number of Rows')

    # rotate x-axis labels
    plt.xticks(rotation=90, ha='right')

    plt.show()

```

```

def top_Authormention(word, quantity):
    # Make a data copy, so the origin of data won't
    changed
    dv1 = dv.copy()

    # Convert the body column to strings to ensure
    that the regular expression search works correctly
    dv1['body'] = dv1['body'].astype(str)

    # Create a new column that contains a boolean
    indicating whether the word appears in the body
    attribute
    dv1['contains_word'] = dv1['body'].apply(lambda
x: True if re.search(r'\b{}\b'.format(word), x,
re.IGNORECASE) else False)

    # Group the dataframe by Author and count the
    number of rows in each group where the word appears
    in the body
    word_counts = dv1[dv1['contains_word'] ==
True].groupby('Author').size().sort_values(ascending=
False)[:quantity]

```

```
# Create the bar plot
fig, ax = plt.subplots()
ax.bar(word_counts.index, word_counts)
ax.set_xlabel('Author')
ax.set_ylabel('Number of Appearances of
("{}").format(word))
ax.set_title('Top {} Authors Mentioning
("{}").format(quantity, word))

# Rotate x-axis labels
plt.xticks(rotation=45, ha='right')

plt.show()
```

```

def top_Authormentionlistofwords(those_words,
quantity): # those_words need to be a list of str
    # Make a data copy, so the origin of data won't
    changed
    dv1 = dv.copy()

    # convert the body column to strings to ensure
    that the regular expression search works correctly
    dv1['body'] = dv1['body'].astype(str)

    # create a new column that contains a boolean
    indicating whether any of the those words appears in
    the body attribute
    dv1['contains_those_words'] =
dv1['body'].apply(lambda x: True if
any(re.search(r'\b{}\b'.format(word), x,
re.IGNORECASE) for word in those_words) else False)

    # group the dataframe by Author, and count the
    number of rows in each group where a those word
    appears in the body
    those_words_counts =
dv1[dv1['contains_those_words'] ==
True].groupby('Author').size().sort_values(ascending=
False)[:10]

```

```

# create the bar plot
fig, ax = plt.subplots()
ax.bar(those_words_counts.index,
those_words_counts)
ax.set_xlabel('Author')
ax.set_ylabel('Number of Appearances of Words
Similat to "{}".format(those_words[0]))
ax.set_title('Top {} Authors by Number of
Appearances of Words Similar to
"{}".format(quantity, those_words[0]))
plt.xticks(rotation=45, ha='right')
plt.show()

```

```

def top_x_score(quantity):
    # Make a data copy, so the origin of data won't
    changed
    dv1 = dv.copy()

    # Sort the dataframe based on the 'score' column
    in descending order
    sorted_dv1 = dv1.sort_values(by=['score'],
ascending=False)

    # Show the top x rows with the highest score
    top_x = sorted_dv1.head(quantity)
    print(top_x)

```

```

def topx_scoreAuthor(quantity):
    # Make a data copy, so the origin of data won't
    changed
    dv1 = dv.copy()

    # Group the dataset by Author and sum the scores
    grouped =
dv1.groupby('Author')['score'].sum().reset_index()

    # Sort the Authors by their total score and take
    the top x
    top_Authors = grouped.sort_values('score',
ascending=False).head(quantity)

    # Plot the graph
    plt.subplots(figsize=(12,6))
    plt.bar(top_Authors['Author'],
top_Authors['score'])
    plt.xlabel('Author')
    plt.ylabel('Combined Score')
    plt.title('Top {} Authors by Combined
Score'.format(quantity))
    plt.xticks(rotation=45, ha='right')
    plt.show()

```

```
def Authorscore(Author):  
    # Make a data copy, so the origin of data won't  
    changed  
    dv1 = dv.copy()  
  
    # Filter the rows by the Author name  
    Author_rows = dv1[dv1['Author'] == Author]  
  
    # Calculate the combined score  
    combined_score = Author_rows['score'].sum()  
  
    # Print the result  
    print(f"The combined score of {Author} is  
{combined_score}.")
```