PEMROGRAMAN JARINGAN

MANUAL BOOK

Web Scraping dan Text Summarization dengan Python dan BART

Disusun Oleh:

Felix Windriyareksa Hardyan (50421506)
4IA12

Ditulis Guna Melengkapi Tugas Pemorgraman Jaringan
Universitas Gunadarma
2025

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Dalam era digital saat ini, informasi dapat diakses dengan sangat mudah melalui berbagai website dan platform online. Mulai dari portal berita, blog, situs ilmiah, hingga ensiklopedia digital seperti Wikipedia, semuanya menyediakan jutaan artikel dan konten yang terus diperbarui setiap hari. Kemudahan ini tentu membawa dampak positif karena informasi menjadi lebih demokratis dan terbuka bagi siapa saja yang membutuhkannya. Namun di sisi lain, kemudahan ini juga menimbulkan tantangan tersendiri, terutama dalam hal bagaimana menyaring dan memahami informasi secara cepat dan efisien.

Volume informasi yang besar sering kali menjadi hambatan bagi pengguna dalam menemukan inti atau pokok bahasan utama dari suatu artikel. Tidak jarang pembaca merasa kewalahan membaca artikel yang panjang, padat, dan penuh dengan detail yang mungkin tidak semuanya relevan. Oleh karena itu, dibutuhkan pendekatan teknologi yang mampu membantu pengguna mengekstrak informasi penting dari sumber digital secara otomatis dan menyajikannya dalam bentuk yang lebih ringkas namun tetap bermakna.

Salah satu solusi dari permasalahan tersebut adalah dengan memanfaatkan teknik Web Scraping, Natural Language Processing (NLP), dan Text Summarization. Web scraping adalah metode otomatisasi untuk mengambil informasi dari halaman web. Dengan teknik ini, pengguna dapat mengakses dan menyalin konten dari situs tertentu tanpa harus melakukannya secara manual.

Setelah konten berhasil diambil, langkah selanjutnya adalah memproses teks menggunakan pendekatan NLP. NLP memungkinkan komputer untuk memahami dan menganalisis bahasa manusia, sehingga teks mentah yang diambil dari web bisa dibersihkan dari kata-kata tidak penting, dipahami strukturnya, dan disiapkan untuk langkah berikutnya.

Langkah terakhir adalah proses Text Summarization, yaitu meringkas informasi panjang menjadi bentuk ringkas yang tetap mencakup ide utama dari teks aslinya. Dengan bantuan model kecerdasan buatan yang canggih, sistem ini dapat menghasilkan

ringkasan secara otomatis dan membantu pengguna memahami konten dalam waktu yang lebih singkat.

Melalui integrasi antara web scraping, NLP, dan text summarization, sistem ini menjadi solusi cerdas untuk merespons tantangan informasi berlimpah di era digital. Selain meningkatkan efisiensi, sistem seperti ini juga dapat diadaptasi untuk berbagai kebutuhan seperti pendidikan, riset, jurnalistik, dan pengembangan produk berbasis data.

1.2. Tujuan

Pada penulisan ini bertujuan utama untuk mempermudah pemahaman dan penerapan pemrograman jaringan dalam membuat program *text summarization.* Adapun tujuan khusus melibatkan:

1. Mengambil konten teks dari web dengan menerapkan *web scraping.*
2. Membersihkan dan memproses teks yang didapatkan dari *web scraping.*
3. Merangkum konten panjang menjadi ringkasan padat (*text summarization*) menggunakan teknologi AI.

BAB II
PEMBAHASAN

2.1. Web Scraping

Web Scraping adalah proses otomatis untuk mengambil informasi dari halaman web. Teknik ini digunakan untuk mengekstrak data mentah dari HTML dan kemudian mengubahnya menjadi informasi yang dapat diolah oleh aplikasi atau sistem. Berikut ini adalah proses cara kerja *web scraping*:

1. **Menentukan Sumber Data (URL *Website*)**

   Memilih alamat URL dari halaman web yang memuat artikel yang ingin dianalisis dan diringkas. Misalnya, pengguna dapat memasukkan URL dari halaman Wikipedia.

2. **Membuat Koneksi ke *Website***

   Setelah URL ditentukan, sistem membuat koneksi ke *website* tersebut menggunakan protokol HTTP atau HTTPS. Dalam Python, hal ini dilakukan menggunakan modul urllib.request, yang berfungsi untuk mengakses konten dari alamat URL yang diberikan oleh pengguna.

3. **Mengambil dan Membaca *Source Code* Halaman *Website***

   Setelah koneksi berhasil, konten halaman web (biasanya dalam format HTML) diunduh dan dibaca. HTML adalah struktur utama yang menyusun tampilan dan isi dari sebuah halaman web.

4. **Menganalisis Struktur HTML untuk Menemukan Bagian yang Diinginkan**

   Konten HTML tersebut kemudian diproses menggunakan *library* **BeautifulSoup** untuk melakukan *parsing* struktur dokumen. Di sini, sistem akan menganalisis elemen HTML seperti `<div>`, `<p>`, atau `<li>` untuk menemukan bagian utama teks artikel. Misalnya, di Wikipedia, bagian utama artikel biasanya berada dalam `<div class="mw-parser-output">`.

5. **Mengekstrak dan Membersihkan Teks**

   Setelah bagian yang relevan ditemukan, teks dari elemen-elemen seperti paragraf dan *list* untuk diekstrak. Teks ini kemudian dibersihkan dari tanda baca, *stopwords* (kata-kata umum yang tidak penting seperti "*the*", "*is*", "*at*"), serta karakter-karakter lain yang tidak diperlukan. Tujuannya agar data lebih bersih dan siap digunakan untuk proses analisis lanjutan.

6. **Menyimpan dan Mengolah Data untuk NLP**

Teks yang telah dibersihkan selanjutnya disimpan dalam format *string* untuk diproses lebih lanjut menggunakan teknik *Natural Language Processing*. Dalam konteks proyek ini, teks tersebut akan diringkas menggunakan model *deep learning* bernama BART (*Bidirectional and Auto-Regressive Transformers***)** dari Facebook AI.

Dengan penerapan *web scraping*, proses pengumpulan informasi dari web tidak lagi dilakukan secara manual, melainkan secara otomatis dan efisien. Data yang sebelumnya tidak terstruktur di halaman web kini bisa diekstrak dan diolah guna membantu pengguna memahami isi artikel secara cepat dan efektif.

2.2. Natural Language Processing (NLP)

Natural Language Processing (NLP) adalah cabang dari Kecerdasan Buatan dan Linguistik, yang didedikasikan untuk membuat komputer memahami pernyataan atau kata-kata yang ditulis dalam bahasa manusia. NLP muncul untuk mempermudah pekerjaan pengguna dan memenuhi keinginan untuk bisa berkomunikasi dengan komputer menggunakan bahasa alami. Karena tidak semua pengguna menguasai bahasa yang spesifik untuk mesin, NLP ditujukan bagi mereka yang tidak memiliki cukup waktu untuk mempelajari bahasa baru atau menjadi mahir dalam bahasa tersebut.

Bahasa dapat didefinisikan sebagai seperangkat aturan atau kumpulan simbol. Simbol-simbol ini dikombinasikan dan digunakan untuk menyampaikan atau menyebarkan informasi. Simbol-simbol tersebut diatur oleh aturan-aturan tertentu. NLP secara umum dapat diklasifikasikan menjadi dua bagian, yaitu *Natural Language Understanding (Linguistics)* dan *Natural Language Generation*, yang mencakup tugas memahami dan menghasilkan teks.

Beberapa tugas NLP yang umum dan banyak diteliti antara lain adalah *Automatic Summarization, Co-Reference Resolution, Discourse Analysis, Machine Translation, Morphological Segmentation, Named Entity Recognition* (NER), *Optical Character Recognition* (OCR), *Part of Speech Tagging* (POS Tagging), dan lain-lain. *Automatic Summarization* menghasilkan ringkasan dari kumpulan teks secara otomatis, sedangkan *Co-Reference Resolution* membantu menentukan kata-kata dalam teks yang merujuk ke objek yang sama. *Discourse Analysis* fokus pada struktur hubungan antar kalimat dalam teks yang berkesinambungan. *Machine Translation* memungkinkan teks diterjemahkan dari satu bahasa manusia ke bahasa lain. *Morphological Segmentation* memecah kata menjadi morfem-morfem dan mengidentifikasi jenisnya. NER

digunakan untuk mengenali nama orang, tempat, atau organisasi dalam teks. OCR memungkinkan pengenalan teks dari gambar, dan POS tagging menentukan kelas kata dari setiap kata dalam kalimat.

2.3. Pengolahan Teks untuk NLP

Pada *Natural Language Processing* (NLP), *text preprocessing* adalah langkah awal yang sangat penting dan memiliki dampak besar terhadap kinerja tugas-tugas lanjutan seperti klasifikasi teks, analisis sentimen, atau penerjemahan mesin. Teks mentah sering mengandung *noise*, inkonsistensi, dan elemen-elemen yang tidak relevan yang perlu dibersihkan atau dinormalisasi sebelum diproses lebih lanjut. Tujuan dari *preprocessing* adalah untuk menstandarisasi dan menyederhanakan teks sambil mempertahankan makna dasarnya. Berikut ini adalah beberapa langkah *text preprocessing* yang sering diterapkan pada NLP:

1. ***Text Cleaning***

   Langkah ini melibatkan penghapusan komponen yang tidak diinginkan dari teks mentah, yaitu:

   a. *Lowercasing*: Mengonversi semua teks menjadi huruf kecil untuk konsistensi dan menghindari perlakuan berbeda antara kata seperti "*Free*" dan "*free*".
   b. *Removing URLs*: Menghapus tautan dari teks karena biasanya tidak memberikan kontribusi berarti pada analisis.
   c. *Removing non-alphanumeric characters*: Menghapus tanda baca, simbol khusus, dan karakter yang tidak relevan yang tidak memberikan nilai tambah.
   d. *Removing numbers*: Menghapus angka karena sering kali tidak memberikan wawasan yang bermakna dalam pemrosesan teks.

2. ***Tokenization***

   Tokenization memecah teks menjadi unit-unit yang lebih kecil, seperti kata atau frasa, sehingga lebih mudah dianalisis dan diproses.

3. ***Stopwords Removal***

   Stopwords adalah kata-kata umum seperti "dan," "itu," "adalah," dan lain-lain yang sering muncul tetapi tidak memberikan banyak makna semantik. Menghapus *stopwords* membantu fokus pada kata-kata yang lebih bermakna.

4. ***Stemming* dan *Lemmatization***

   a. *Stemming*: mengurangi kata-kata ke bentuk dasarnya dengan menghapus prefix dan *suffix*, misalnya "*running*" menjadi "*run*."

b. *Lemmatization*: Mengubah kata-kata ke bentuk dasar berdasarkan makna dan bagian dari kata tersebut, misalnya "*better*" menjadi "*good*."

2.4. BART (Bidirectional and Auto-Regressive Transformers)

Bidirectional and Auto-Regressive Transformers (BART) adalah model yang dikembangkan oleh Facebook AI. BART adalah sebuah *denoising autoencoder* yang menerapkan paradigma *sequence-to-sequence*, menjadikannya sangat efektif untuk berbagai aplikasi. Proses pretraining BART terdiri dari dua tahap: (1) teks "dirusak" dengan menggunakan fungsi *noising* yang bersifat arbitrer, dan (2) model *sequence-to-sequence* dilatih untuk merekonstruksi teks asli dari teks yang telah dirusak.

Selain kekuatannya dalam tugas pemahaman teks, efektivitas BART semakin meningkat ketika dilakukan fine-tuning untuk tugas text generation. Model ini menghasilkan kinerja terbaik pada berbagai tugas abstractive conversation, question answering, dan summarization, bahkan mampu mencapai performa setara dengan RoBERTa dengan sumber daya pelatihan yang sebanding pada GLUE dan SQuAD.

BART mengadopsi arsitektur Transformer *sequence-to-sequence* standar sebagaimana dijelaskan oleh Vaswani et al. (2017), dengan beberapa modifikasi. Mengikuti pendekatan GPT, BART menggantikan fungsi aktivasi ReLU dengan GeLUs (Hendrycks & Gimpel, 2016) dan menginisialisasi parameter dengan distribusi N (0, 0.02). Untuk model versi *base*, BART menggunakan 6 lapisan pada *encoder* dan *decoder*, sementara model versi *large* menggunakan 12 lapisan pada masing-masing komponen. Arsitektur BART sangat mirip dengan arsitektur BERT, dengan perbedaan utama sebagai berikut: (1) setiap lapisan decoder melakukan *cross-attention* terhadap lapisan tersembunyi terakhir dari encoder, seperti halnya pada model *sequence-to-sequence* Transformer; dan (2) BERT menyertakan jaringan *feed-forward* tambahan sebelum tahap prediksi kata, sementara BART tidak. Secara keseluruhan, BART memiliki sekitar 10% lebih banyak parameter dibandingkan dengan model BERT yang memiliki ukuran serupa.

BAB III
ANALISA DAN PERANCANGAN

3.1. Instalasi dan *Import* Library

Sebelum menjalankan program, instalasi library eksternal diperlukan. Library utama yang digunakan antara lain:

1. **beautifulsoup4** dan **lxml**: digunakan untuk *scraping* dan *parsing* HTML.
2. **nltk**: digunakan untuk *preprocessing* teks seperti menghapus *stopwords*.
3. **transformers** dan **torch**: digunakan untuk menjalankan model BART untuk *summarization*.

```
!pip install beautifulsoup4 lxml nltk transformers torch requests
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.5 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 36.9 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 41.8 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 40.3 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.4 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.0 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 11.3 MB/s eta 0:00:00
```

Setelah itu, *library* yang dibutuhkan di-*import* menggunakan kode berikut:

```
import bs4 as bs
import urllib.request
import string
import nltk
from nltk.corpus import stopwords
from transformers import pipeline, AutoTokenizer
```

```
# Download stopwords
nltk.download("stopwords")
```

Penjelasan kode:

1. `bs4 as bs`: memanggil BeautifulSoup untuk *parsing* HTML.
2. `urllib.request`: membuka URL dan membaca kontennya.
3. `string`: menyediakan daftar tanda baca (untuk dihapus).
4. `nltk` dan `stopwords`: *tools* dari NLTK untuk *text preprocessing*.

5. `pipeline, AutoTokenizer`: *class* dari *transformers* untuk menjalankan dan mempersiapkan model *summarization*.
6. `nltk.download("stopwords")`: mengunduh daftar *stopwords*, seperti "*and*", "*the*", "*is*" dari *library* NLTK.

3.2. *Web Scraping*

Tahap ini berfungsi untuk mengambil isi artikel dari sebuah URL. Program meminta pengguna memasukkan URL, kemudian menggunakan **urllib** dan **BeautifulSoup** untuk mengambil dan mem-*parsing* konten HTML-nya. Pertama-tama program akan meminta *user* untuk memasukkan URL yang ingin di-*scrape*:

```
# Input URL dari pengguna
url = input("Please input the URL you want to scrape: ")
```

```
Please input the URL you want to scrape: https://www.bbc.com/sport/formula1/articles/c78egyj4ng3o
```

Kemudian program akan membuka URL, membaca konten web tersebut, dan *parsing* HTML menggunakan library **BeautifulSoup**:

```
web_scraping = urllib.request.urlopen(url)
content = web_scraping.read()
parsing = bs.BeautifulSoup(content, 'lxml')
```

Program mencari bagian utama teks artikel. Jika `div` dengan *class* `mw-parser-output` tersebut tidak ditemukan, *fallback*-nya adalah semua elemen `<p>` dengan kode berikut:

```
# Mengambil teks utama dari website
content_div = parsing.find('div', {'class': 'mw-parser-output'})
paragraphs = content_div.find_all(['p', 'li']) if content_div else parsing.find_all('p')
```

Terakhir semua teks dari elemen paragraf dan list akan digabungkan menjadi satu *string*:

```
# Gabungkan seluruh teks artikel
article_text = " ".join([p.text for p in paragraphs])
```

3.3. *Preprocessing* Teks

Setelah teks di ekstrak pada proses *web scraping*, teks tersebut perlu diolah terlebih dahulu agar siap digunakan. Berikut ini adalah tahapan *preprocessing* yang dilakukan:

```
# Fungsi untuk membersihkan teks
def clean_text(text):
    if not text:
        return ""
    text = ''.join([char for char in text if char not in string.punctuation])
    words = [word for word in text.split() if word.lower() not in stopwords.words('english')]
    return ' '.join(words)

# Bersihkan teks
cleaned_text = clean_text(article_text)
```

Penjelasan langkah-langkah jalannya fungsi `clean_text()`:

1. Mengecek apakah teks kosong.
2. Menghapus tanda baca (.,!?) dengan menggunakan `string.punctuation`
3. Memecah teks menjadi kata menggunakan `text.split()`
4. Menghapus kata-kata yang termasuk stopwords (`not in stopwords.words('english')`).
5. Menggabungkan kembali kata menjadi teks bersih.

Kemudian setelah membuat fungsi `clean_text()`, program akan melakukan dua pengecekan. Pertama, jika teks hasil pembersihan kosong, program akan keluar. Kedua, jika teks terlalu pendek, program juga akan dihentikan. BART butuh input yang cukup panjang agar bisa memberi ringkasan yang baik. Pengecekan dilakukan dengan kode berikut:

```
if not cleaned_text:
    print("Teks kosong setelah dibersihkan. Tidak dapat diringkas.")
    exit()

if len(cleaned_text.split()) < 50:
    print("Teks terlalu pendek untuk diringkas. Gunakan artikel yang lebih panjang.")
    exit()
```

Panjang teks juga perlu dibatasi, karena BART memiliki batas maksimal token input sebanyak 1024. Jika teks terlalu panjang maka hanya 1024 kata yang digunakan. Berikut adalah kodenya:

```
max_words = 1024
word_list = cleaned_text.split()
if len(word_list) > max_words:
    cleaned_text = " ".join(word_list[:max_words])
```

3.4. Tokenisasi Teks dan *Modeling*

Teks yang sudah dibersihkan kemudian dapat ditokenisasi berikut ini adalah kode yang digunakan untuk tokenisasi teks:

```
# Memuat model summarization dan tokenizer menggunakan AutoTokenizer
model_name = "facebook/bart-large-cnn"
summarizer = pipeline("summarization", model=model_name)  # Memuat pipeline untuk merangkum teks
tokenizer = AutoTokenizer.from_pretrained(model_name)  # Memuat tokenizer untuk model
```

Pertama model `facebook/bart-large-cnn` untuk dipilih untuk task *text summarization*. Kemudian `pipeline` dari Hugging Face digunakan untuk otomatisasi *input/output summarization*. `AutoTokenizer` dipilih agar proses tokenisasi dilakukan secara otomatis sesuai dengan model yang dipilih. Data teks yang telah di tokensasi kemudian dapat digunakan pada model BART dengan menggunakan kode berikut:

```
# Lakukan summarization
summary = summarizer(cleaned_text, max_length=150, min_length=50, do_sample=False)
```

BART akan membuat ringkasan teks dengan `max_length=150` (maksimal token hasil ringkasan), `min_length=50` (minimal token hasil ringkasan). Agar hasil ringkasan selalu sama untuk input yang sama (deterministik), `do_sample=False` digunakan.

3.5. *Output* Ringkasan Teks

Sebelum menampilkan hasil ringkasan teks, pembuatan *function* `wrap_text()` diperlukan agar hasil teks ditampilkan lebih rapih dan melebar dalam satu baris panjang. Berikut adalah kode yang digunakan:

```
import textwrap

# Fungsi untuk wrap teks
def wrap_text(text, width=80):
    return "\n".join(textwrap.wrap(text, width=width))

# Output ringkasan
print("\nOriginal Text:\n", wrap_text(article_text[:1000]))
print("\nSummary:\n", wrap_text(summary[0]['summary_text']))
```

Function ini menggunakan library textwrap untuk memotong text menjadi beberapa baris dengan panjang maksimal 80 karakter. Kemudian teks original dan teks hasil ringkasan akan ditampilkan pada output program:

```
Original Text:
 Seven-time champion Hamilton won his first ever sprint race on Saturday Lewis
Hamilton hit out at "yapping" critics after taking his first win for Ferrari in
the sprint race at the Chinese Grand Prix. The seven-time champion followed up
his win in only his second event for his new team with fifth place on the grid
for Sunday's main event but said he was "optimistic" of a good result. Hamilton
did not identify the people he was referring to but said they "lacked
understanding" of how difficult it was to achieve success straight away with a
new team. The 40-year-old said: "People just love to be negative at any
opportunity. Even with the smallest things, they'll just be negative about it.
"That's just the difficult time that we're living in. "I see certain individuals
– and again, I don't read the news, but I see bits here and there – see people
that I've admired for years just talking out of turn. "Clearly some of them
really just making uneducated guesses of what's going on, just a rea

Summary:
 Lewis Hamilton hit yapping critics taking first win Ferrari sprint race Chinese
Grand Prix. Seventime champion Hamilton first ever sprint race Saturday.
Hamilton teammate Charles Leclerc together third row Piastri took pole Mercedes.
George Russell Lando Norris Australia McLaren Verstappen fourth grid grand prix
ahead Hamilton.
```

3.6. File Project dan Video Demonstrasi

File project dan video demonstrasi program *Web Scraping* dan *Text Summarization* dapat diakses pada link Google Drive di bawah ini: https://drive.google.com/drive/folders/1zPjyQEL95csFoTTAs5AtxncXZdXbLcOt?usp=sharing

BAB IV
PENUTUP

4.1. Kesimpulan

Web Scraping adalah proses otomatis untuk mengambil informasi dari halaman web. Teknik ini digunakan untuk mengekstrak data mentah dari HTML dan kemudian mengubahnya menjadi informasi yang dapat diolah oleh aplikasi atau sistem. *Natural Language Processing* (NLP) adalah bidang dalam kecerdasan buatan (AI) yang berfokus pada interaksi antara komputer dan bahasa manusia.

Bidirectional and Auto-Regressive Transformers (BART) adalah model yang dikembangkan oleh Facebook AI. BART memiliki arsitektur *encode-decoder transformer*, *encoder* berfungsi untuk memproses input teks dan memahami konteks teks, sedangkan *decoder* berfungsi untuk mennghasilkan teks baru berdasarkan pemahaman yang didapatkan dari *encoder*, misalnya membuat ringkasan.

Proyek ini merupakan implementasi lengkap untuk melakukan *web scraping* artikel, dilanjutkan dengan *preprocessing* teks, dan kemudian diringkas menggunakan model BART (facebook/bart-large-cnn) dari Hugging Face. Proses ini terdiri dari enam tahapan utama: instalasi *library*, *scraping* konten HTML, pembersihan teks, tokenisasi, penerapan model *summarization*, serta menampilkan ringkasan dengan format yang rapi. Sistem ini memungkinkan *user* untuk memperoleh ringkasan otomatis dari artikel secara efisien, dengan hanya memasukkan URL halaman yang diinginkan.