PENERAPAN DAN EVALUASI MODEL *BIDIRECTIONAL AND AUTO-REGRESSIVE TRANSFORMER* (BART) DALAM PERINGKASAN TEKS ABSTRAKTIF ARTIKEL ILMIAH

TUGAS AKHIR

Angelica Noviana
121450064

ITERA

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

PENERAPAN DAN EVALUASI MODEL *BIDIRECTIONAL AND AUTO-REGRESSIVE TRANSFORMER* (BART) DALAM PERINGKASAN TEKS ABSTRAKTIF ARTIKEL ILMIAH

TUGAS AKHIR

Diajukan sebagai syarat untuk memperoleh gelar Sarjana

Angelica Noviana
121450064

ITERA

PROGRAM STUDI SAINS DATA
FAKULTAS SAINS
INSTITUT TEKNOLOGI SUMATERA
2025

HALAMAN PENGESAHAN

Tugas Akhir Sarjana dengan judul "**Penerapan dan Evaluasi Model *Bidirectional and Auto-Regressive Transformer* (BART) dalam Peringkasan Teks Abstraktif Artikel Ilmiah**" adalah benar dibuat oleh saya sendiri dan belum pernah dibuat dan diserahkan sebelumnya, baik sebagian ataupun seluruhnya, baik oleh saya ataupun orang lain, baik di Institut Teknologi Sumatera maupun di institusi pendidikan lainnya.

Lampung Selatan, 02 September 2025

Penulis,

Angelica Noviana
NIM. 121450064

Diperiksa dan disetujui oleh,

Pembimbing I	Pembimbing II
Tirta Setiawan, S.Pd., M.Si	**Ade Lailani, M.Si**
NIP. 199008222022031003	**NRK. 1996070120232278**

Disahkan oleh,
Koordinator Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

Tirta Setiawan, S.Pd., M.Si
NIP. 199008222022031003

Sidang Tugas Akhir:

Penguji I : Christyan Tamaro Nadeak, M.Si
Penguji II : Yusni Puspha Lestari, S.T., M.Si

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah karya saya sendiri dan semua sumber baik yang dikutip maupun yang dirujuk telah saya nyatakan benar.

Nama	:	**Angelica Noviana**
NIM	:	**121450064**
Tanda tangan	:	
Tanggal	:	**02 September 2025**

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Institut Teknologi Sumatera, saya yang bertanda tangan di bawah ini:

Nama	:	Angelica Noviana
NIM	:	121450064
Program Studi	:	Sains Data
Fakultas	:	Sains
Jenis karya	:	Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan Hak Bebas Royalti Noneksklusif (*Non-Exclusive Royalty Free Right*) kepada Institut Teknologi Sumatera atas karya ilmiah saya yang berjudul:

Penerapan dan Evaluasi Model *Bidirectional and Auto-Regressive Transformer* (BART) dalam Peringkasan Teks Abstraktif Artikel Ilmiah

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Sumatera berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di	:	Lampung Selatan
Pada tanggal	:	02 September 2025

Yang menyatakan

Angelica Noviana

ABSTRAK

Penerapan dan Evaluasi Model *Bidirectional and Auto-Regressive Transformer* (BART) dalam Peringkasan Teks Abstraktif Artikel Ilmiah

Angelica Noviana (121450064)
Pembimbing I: Tirta Setiawan, S.Pd., M.Si
Pembimbing II: Ade Lailani, M.Si

Penelitian ini bertujuan mengevaluasi performa model *Bidirectional and Auto-Regressive Transformer* (BART) dalam menghasilkan ringkasan abstraktif untuk artikel ilmiah bertema *"data science"*. Sebelum pelatihan model, dilakukan analisis distribusi kata pada data mentah hasil *scraping* dari arXiv dan data yang telah melalui tahap *preprocessing*, termasuk frekuensi kata, visualisasi *Word Cloud*, dan distribusi panjang kalimat. Hasil analisis menunjukkan bahwa *preprocessing* berperan penting dalam meningkatkan representasi data. Data kemudian dibagi menjadi data latih dan uji (80:20), dan model dilatih menggunakan kombinasi *hyperparameter* tertentu. Evaluasi dilakukan secara kuantitatif menggunakan metrik ROUGE dan secara kualitatif melalui interpretasi ringkasan. Konfigurasi terbaik mencapai ROUGE-1 sebesar 0.60, ROUGE-2 sebesar 0.52, dan ROUGE-L sebesar 0.56. Saat diuji pada artikel baru, diperoleh ROUGE-1 sebesar 0.48, ROUGE-2 sebesar 0.36, dan ROUGE-L sebesar 0.43. Hasil ringkasan menunjukkan kemampuan model dalam menangkap inti pembahasan artikel, meskipun beberapa detail teknis belum tergambarkan. Penelitian ini menunjukkan potensi BART untuk diterapkan lebih lanjut dalam otomatisasi peringkasan teks ilmiah.

Kata kunci: Peringkasan Teks, Ringkasan Abstraktif, BART, Skor ROUGE, Artikel Ilmiah

ABSTRACT

Implementation and Evaluation of the Bidirectional and Auto-Regressive Transformer (BART) Model for Abstractive Summarization of Scientific Articles

Angelica Noviana (121450064)
Advisor I: Tirta Setiawan, S.Pd., M.Si
Advisor II: Ade Lailani, M.Si

This research evaluates the performance of the Bidirectional and Auto-Regressive Transformer (BART) model in generating abstractive summaries of scientific articles focused on the topic of data science. Prior to model training, the dataset collected from arXiv via web scraping was analyzed in both raw and preprocessed forms, including word frequency, Word Cloud visualization, and sentence length distribution. The analysis showed that preprocessing significantly improved data representation. The dataset was split into training and test sets (80:20), and the model was trained using selected hyperparameter combinations. Evaluation was conducted using ROUGE metrics and supported by qualitative interpretation. The best configuration achieved ROUGE-1 of 0.60, ROUGE-2 of 0.52, and ROUGE-L of 0.56. When tested on new articles, it produced ROUGE-1 of 0.48, ROUGE-2 of 0.36, and ROUGE-L of 0.43. The generated summaries demonstrated the model's ability to capture the main content of the articles, though some technical details were not fully represented. These findings highlight BART's potential for further applications in scientific text summarization.

Keywords : *Text Summarization, Abstractive Summarization, BART, ROUGE Score, Scientific Article*.

MOTTO

"Don’t stop when you’re tired. Stop when you’re done!"

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur, karya ini saya persembahkan kepada orang tua tercinta atas doa dan kasih sayang yang tiada henti, keluarga yang selalu menjadi sumber kekuatan, serta sahabat dan rekan seperjuangan yang senantiasa memberikan dukungan dan semangat. Tak lupa, persembahan ini juga untuk semua pihak yang dengan cara masing-masing telah memberikan inspirasi hingga terselesaikannya karya sederhana ini.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat-Nya, sehingga penulis dapat menyelesaikan penyusunan tugas akhir ini dengan baik. Tugas akhir ini disusun untuk memenuhi salah satu syarat guna memperoleh gelar Sarjana Sains Data pada Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera. Dalam proses penyusunan tugas akhir ini, penulis banyak menerima bantuan, bimbingan, serta dukungan dari berbagai pihak. Oleh karena itu, dengan kerendahan hati penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua serta keluarga yang selalu memberikan doa, semangat, dan kasih sayang tanpa henti.
2. Bapak Tirta Setiawan, S.Pd., M.Si, selaku koordinator program studi yang sudah memudahkan penulis dalam pengetahuan alur waktu penyusunan tugas akhir, memberikan arahan, ilmu, dan masukan berharga selama proses penelitian maupun penulisan, serta menjadi dosen pembimbing I penulis dalam penyusunan tugas akhir.
3. Ibu Ade Lailani, M.Si, selaku dosen pembimbing II yang telah membimbing, mengarahkan, dan memberikan saran dalam penyusunan tugas akhir ini.
4. Bapak Christyan Tamaro Nadeak, M.Si, selaku dosen penguji I yang telah memberikan masukan, kritik, serta saran yang sangat berharga dalam penelitian ini, sehingga penulisan tugas akhir dapat diperbaiki dan disempurnakan.
5. Ibu Yusni Puspha Lestari, S.T., M.Si, selaku dosen penguji II yang telah memberikan arahan dan koreksi yang membangun, sehingga penulis dapat memperbaiki kelemahan dan menyelesaikan tugas akhir ini dengan lebih baik.

6. Bapak/Ibu dosen Program Studi Sains Data yang telah memberikan ilmu pengetahuan, wawasan, dan pengalaman akademik yang sangat berarti.
7. Rekan-rekan mahasiswa Sains Data angkatan 2021 yang selalu memberikan semangat, bantuan, serta kebersamaan yang sangat berharga. Ucapan terima kasih juga penulis sampaikan kepada Sella, Dwi, Intan, Aisyah Tiara, Araaa, dan teman-teman lainnya atas dukungan dan bantuan yang diberikan dalam menyelesaikan tugas akhir ini.
8. Semua pihak yang tidak dapat penulis sebutkan satu per satu, namun telah memberikan dukungan baik secara langsung maupun tidak langsung.

Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa yang akan datang. Semoga tugas akhir ini dapat memberikan manfaat, baik bagi penulis maupun bagi pembaca yang memerlukannya.

Lampung Selatan, 02 September 2025

Angelica Noviana

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SIMBOL

$\mathcal{L}_{DAE}(X, X')$	:	Fungsi *loss denoising autoencoder* pada BART
$P(X_i \mid X')$	:	Probabilitas model untuk memprediksi token asli
$\sum_{j=1}^{n} \exp(e_j)$	:	Total skor token j ke semua token
ROUGE-N	:	Skor ROUGE untuk n-gram tertentu
ROUGE-L	:	Skor ROUGE berdasarkan LCS
LCS	:	Panjang subsekuens yang sama dari kedua ringkasan
X	:	Urutan token asli (teks asli) sebelum dimodifikasi
X'	:	Urutan token yang telah dimodifikasi
n	:	Jumlah token dalam teks
α_l	:	Bobot perhatian (*attention weight*) pada posisi ke-l
e_l	:	*Attention score* antara kata pada posisi ke-l
Q_i	:	Vektor *query* pada posisi i
K_j	:	Vektor *key* pada posisi j
d_k	:	Dimensi vektor *query/key* untuk normalisasi
c	:	*Context vector* sebagai *output Attention Mechanism*
v_l	:	*Value vector* ke-l
p	:	Jumlah n-gram yang sama dari kedua ringkasan
q	:	Jumlah total n-gram dalam ringkasan referensi
m	:	Jumlah kata dalam ringkasan referensi

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi telah mempercepat penyebaran informasi seperti berita, artikel, dan iklan yang kini dapat diakses secara *online* melalui internet. Dalam konteks penyebaran artikel ilmiah, saat ini tersedia berbagai repositori daring yang menyimpan serta menyediakan akses ke publikasi akademik dari beragam disiplin ilmu. Beberapa contoh repositori tersebut adalah *Elsevier, IEEE Xplore*, dan *Wiley Online Library* yang sudah sangat dikenal di kalangan akademisi dan peneliti. Meskipun repositori-repositori ini menawarkan akses melalui *Application Programming Interface* (API), layanan tersebut umumnya memiliki sejumlah keterbatasan sehingga menyulitkan pengambilan data dalam jumlah besar. *Elsevier* membatasi akses berdasarkan langganan institusi dan jumlah permintaan mingguan [1]. *IEEE Xplore* mensyaratkan penggunaan akun institusi dan memiliki batas permintaan API [2]. Sementara *Wiley Online Library* hanya memberikan akses kepada pengguna berlisensi *Text and Data Mining* (TDM) [3].

Berbeda dari ketiganya, terdapat repositori akses terbuka yang menyediakan artikel ilmiah secara gratis untuk berbagai kebutuhan penelitian. Repositori seperti ini biasanya juga menyediakan antarmuka pemrograman aplikasi terbuka (*open API*) yang mendukung ekstraksi data otomatis. Salah satu contohnya adalah arXiv, dengan lebih dari 2.5 juta artikel, mendukung publikasi cepat sebelum *peer review*, dan cocok untuk penelitian berbasis teks serta *Natural Language Processing* (NLP) [4], [5]. Seiring bertambahnya jumlah publikasi ilmiah, tantangan untuk menemukan jurnal yang

relevan pun semakin meningkat. Oleh karena itu, diperlukan metode peringkasan teks agar pembaca dapat dengan mudah memperoleh informasi penting tanpa harus membaca keseluruhan dokumen.

Meskipun setiap artikel ilmiah telah dilengkapi dengan abstrak, perbedaan dalam detail informasi, panjang teks, susunan, dan gaya penulisan abstrak setiap artikel tidak selalu efektif dalam menyampaikan inti artikel secara konsisten. Berdasarkan penelitian Iana dkk. (2016), sebanyak 84% abstrak artikel memiliki kalimat yang sama persis dengan isi artikelnya. Kondisi ini menunjukkan bahwa sebagian besar abstrak hanya menyalin dari teks aslinya, sehingga cenderung kurang informatif. Hal tersebut mempertegas perlunya pengembangan metode peringkasan teks yang lebih efektif [6]. Terdapat 2 jenis peringkasan teks, yaitu peringkasan ekstraktif yang memilih dan menyalin bagian penting dari teks asli dengan mempertimbangkan seberapa sering kata muncul atau seberapa relevan kalimat tersebut mewakili dokumen, dan peringkasan abstraktif yang menghasilkan ringkasan dengan menyusun ulang informasi menggunakan kalimat baru yang lebih singkat dan alami [7]. Peringkasan abstraktif semakin populer karena kemampuannya menghasilkan ringkasan yang lebih alami, kreatif, dan relevan dengan teks aslinya [8]. Berbagai metode berbasis *machine learning* dan *deep learning* telah banyak dikembangkan untuk mendukung efektivitas peringkasan abstraktif yang lebih alami dan akurat.

Deep learning adalah cabang dari *machine learning* yang memodelkan abstraksi data melalui fungsi non-linier dalam beberapa lapisan [9]. Minat terhadap *neural network* meningkat berkat kemajuan teknologi *big data* dan pemrosesan GPU yang memungkinkan pengolahan data besar secara efisien [9]. Evaluasi

kualitas ringkasan teks biasanya menggunakan metrik *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE), yang mencakup ROUGE-1 untuk mengukur kesamaan ringkasan berdasarkan *unigram*, ROUGE-2 untuk mengukur kesamaan *bigram*, serta ROUGE-L yang mengukur kesamaan ringkasan berdasarkan subsekuens kata terpanjang yang sama (*longest common subsequence*) [10]. Penelitian oleh Shafiq dkk. (2023) menunjukkan bahwa model *deep learning* seperti *recurrent neural network* (RNN) untuk peringkasan teks bahasa Urdu lebih efektif dibandingkan metode *machine learning* tradisional seperti *Support Vector Machine* (SVM) dan *logistic regression* [11]. Model RNN mencapai presisi 28,14% pada ROUGE-1 dan 28,49% pada ROUGE-L, sedangkan SVM mencapai ROUGE-1 sebesar 21,28% dan ROUGE-2 sebesar 11,46%, dan *logistic regression* mencapai ROUGE-1 sebesar 19,65% dan ROUGE-2 sebesar 9,75% [11]. Hasil penelitian tersebut menunjukkan bahwa model berbasis *deep learning* memiliki kemampuan yang baik dalam menghasilkan ringkasan ilmiah, sehingga mendorong pemanfaatan model dengan arsitektur canggih seperti *Bidirectional and Auto-Regressive Transformer* (BART) pada penelitian ini.

Model BART adalah model *pre-training* berbasis *transformer*, yaitu arsitektur yang dapat memperhatikan bagian-bagian penting dalam data berurutan, sehingga bisa memproses informasi dengan lebih cepat dan efisien [12]. BART menggabungkan *encoder bidirectional* yang membaca teks dari dua arah untuk memahami konteks teks secara menyeluruh, dan *decoder autoregresif* yang menghasilkan teks secara satu per satu berdasarkan kata-kata sebelumnya untuk mendapatkan ringkasan yang lebih alami [13]. BART juga menggunakan 5 teknik *pre-training* dalam bentuk *denoising autoencoder*, yaitu metode pelatihan yang membuat

model belajar memulihkan teks yang diubah atau diacak seperti *token masking* (menyembunyikan kata), *sentence permutation* (mengacak urutan kalimat), *document rotation* (mengubah posisi awal teks), *token deletion* (menghapus kata), dan *text infilling* (mengisi bagian kosong dalam teks) [13]. Penelitian oleh La Quatra M dkk. (2023) menunjukkan efektivitas BART dalam peringkasan teks bahasa Italia dengan skor ROUGE-1 42,32, ROUGE-2 28,83, dan ROUGE-L 38,84 [14]. Penelitian ini memanfaatkan model BART untuk peringkasan abstraktif artikel ilmiah di *platform* arXiv melalui proses *scraping* data pada topik "*data science*". Tujuannya adalah mengoptimalkan model BART untuk menghasilkan ringkasan ilmiah yang akurat, terstruktur, dan mudah dipahami.

1.2 Rumusan Masalah

Adapun rumusan masalah penelitian ini, sebagai berikut :

1. Apa karakteristik distribusi kata dalam artikel ilmiah pada topik *"data science"*?
2. Bagaimana performa model BART dalam menghasilkan ringkasan teks abstraktif terhadap artikel ilmiah berdasarkan evaluasi ROUGE *Score* dan analisis kualitas ringkasan?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut :

1. Mengidentifikasi karakteristik distribusi kata dalam artikel ilmiah pada topik *"data science"*.
2. Mengevaluasi performa model BART dalam menghasilkan ringkasan teks abstraktif artikel ilmiah menggunakan ROUGE *Score* dan analisis kualitas isi ringkasan yang dihasilkan.

1.4 Batasan Masalah

Batasan permasalahan pada penelitian ini adalah sebagai berikut :

1. Penelitian ini hanya menggunakan BART untuk menghasilkan ringkasan teks abstrak. Tidak ada model lain yang digunakan dalam cakupan penelitian ini.
2. Dataset yang digunakan hanya berasal dari ArXiv dengan topik terkait "*data science*", dari rentang Tahun 2014 - 2024.
3. Berfokus pada *abstractive summarization*, yaitu menghasilkan ringkasan dengan kata-kata yang berbeda dari teks asli.

BAB II

TINJAUAN PUSTAKA

2.1 Studi Literatur

Penelitian terdahulu terkait BART dan peringkasan teks abstraktif yang menjadi acuan dalam penelitian ini ditunjukkan dalam Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu *Summarization*

Tahun	Penulis	Judul	Metode	Hasil
2024	Pascal Wilmana, dkk.	*Abstractive English Document Summarization Using BART Model with Chunk Method*	BART	Evaluasi ROUGE *Score* menunjukkan bahwa metode *chunking* pada BART lebih efektif untuk ringkasan dokumen panjang, dengan skor ROUGE-1: 42.15, ROUGE-2: 19.87, dan ROUGE-L: 38.42 [15].
2022	Moreno La Quatra, dkk.	*BART-IT: An Efficient Sequence-to-Sequence Model for Italian Text Summarization*	BART-IT	Model ini menunjukkan performa baik dengan ROUGE-1: 35.42, ROUGE-2: 15.88, dan ROUGE-L: 25.12, meski masih perlu perbaikan untuk konteks bahasa yang kompleks [16].
2024	Gaduh Hartawan, dkk.	*Bidirectional and Auto-Regressive Transformer (BART) for Indonesian Abstractive Text Summarization*	BART	Model BART menunjukkan kemampuan dalam menghasilkan ringkasan yang baik dan relevan, dengan skor ROUGE-1 sebesar 37.19, ROUGE-2 sebesar 14.03, dan ROUGE-L sebesar 33.85 [10].

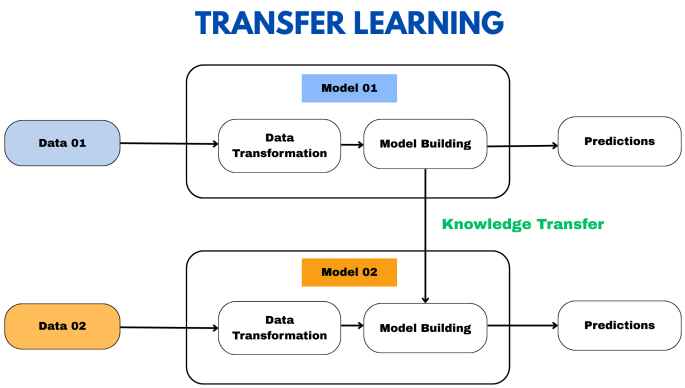
2.2 *Summarization*

Summarization atau peringkasan teks bertujuan untuk menghasilkan ringkasan dari teks panjang secara otomatis [17]. Proses ini memungkinkan pengguna memperoleh informasi penting secara efisien [17]. Peringkasan teks otomatis dapat dibedakan menjadi peringkasan abstraktif dan ekstraktif [18]. Peringkasan abstraktif menggunakan kata-kata baru tanpa mengambil langsung dari teks asli [18]. Peringkasan ini menciptakan ringkasan yang lebih mudah dipahami dengan tata bahasa yang lebih sederhana [18]. Sementara itu, peringkasan ekstraktif akan memilih dan menyalin bagian penting dari teks asli dengan mempertimbangkan seberapa sering kata muncul atau seberapa relevan kalimat tersebut mewakili dokumen, dan kata-kata dengan skor tertinggi akan digabungkan untuk membentuk ringkasan [18]. Penelitian oleh Kirmani dkk. (2024) melakukan analisis terhadap berbagai metode peringkasan teks baik ekstraktif maupun abstraktif, dengan mengevaluasi efektivitas metode ekstraktif seperti frekuensi kata dan posisi kalimat, serta metode abstraktif berbasis *neural network* yaitu model yang meniru cara kerja otak manusia dalam membuat keputusan [19]. Penelitian tersebut menunjukkan bahwa meskipun pendekatan ekstraktif lebih efisien secara komputasi, tetapi metode abstraktif menghasilkan ringkasan yang lebih koheren, *natural*, dan kontekstual sehingga lebih unggul dalam kualitas *output* [19].

2.3 *Transfer Learning*

Transfer learning adalah pendekatan dalam *machine learning* yang memanfaatkan model yang telah dilatih pada dataset awal untuk diterapkan pada dataset baru yang relevan [20]. Tujuannya adalah untuk meningkatkan kinerja model pada dataset kecil atau terbatas dengan memanfaatkan pengetahuan dari dataset yang lebih besar,

serta meningkatkan efisiensi pelatihan dengan menggunakan representasi fitur yang telah dipelajari sebelumnya [20]. Konsep *transfer learning* sangat penting dalam NLP, di mana model seperti BART, yang merupakan salah satu arsitektur *transformer* populer, awalnya dilatih pada data berukuran besar lalu disesuaikan lebih lanjut untuk tugas-tugas khusus seperti *text summarization, question answering*, dan *text classification* [21]. Dalam *text summarization*, *transfer learning* dapat meningkatkan kemampuan generalisasi model, mengurangi kebutuhan data pelatihan besar dan mempercepat pelatihan [22]. Contoh konsep *transfer learning* terdapat pada Gambar 2.1 [23].



Gambar 2.1 Konsep *transfer learning* (digambar ulang oleh penulis)

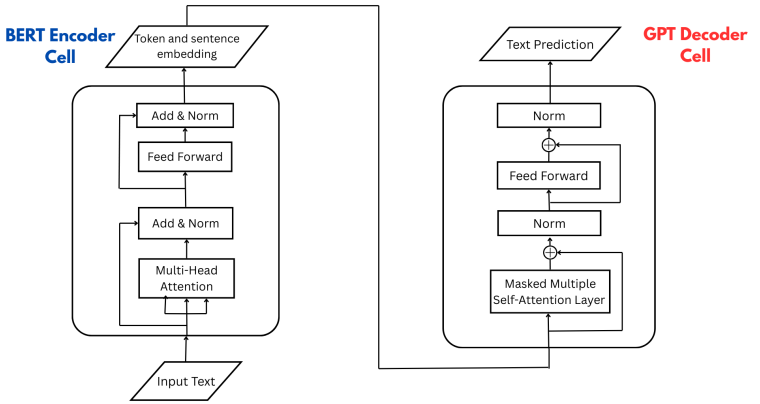
2.4 BART

Bidirectional and Auto-Regressive Transformer (BART) adalah model berbasis *sequence-to-sequence* (*seq2seq*) yaitu metode yang terdiri dari *encoder* dan *decoder*, di mana masing-masing model memproses *input* dan menghasilkan *output* dalam bentuk vektor [24]. BART menggunakan pendekatan *autoencoder denoising*, yaitu model yang dilatih untuk memulihkan teks yang telah diubah atau diacak ke bentuk aslinya, sehingga mampu memahami struktur

dan makna teks secara lebih baik [25]. BART menggabungkan keunggulan BERT (*Bidirectional Encoder Representations from Transformers*) yang membaca teks dari dua arah untuk memahami konteks teks secara menyeluruh, dan GPT (*Generative Pre-trained Transformer*) yang menghasilkan teks secara satu per satu berdasarkan kata-kata sebelumnya, dalam arsitektur berbasis *transformer* yang sederhana namun efektif. Model ini dilatih menggunakan fungsi *loss denoising autoencoder* yang menghitung seberapa baik model memulihkan teks yang dimodifikasi melalui teknik seperti *masking* (menyembunyikan/mengganti kata dengan tanda khusus) [25]. *Transformer* adalah model yang menggunakan cara khusus untuk memperhatikan bagian-bagian penting dalam data berurutan, sehingga bisa memproses informasi lebih cepat dan efektif [12]. BART juga memanfaatkan mekanisme *self-attention* yang memungkinkan setiap kata dalam kalimat memperhatikan kata-kata lainnya, sehingga mampu memahami hubungan antar token dan menghasilkan ringkasan serta rekonstruksi teks yang akurat [26]. Alur kerja model BART dapat dilihat pada Gambar 2.2, dengan Persamaan (2.1) untuk menghitung fungsi *loss* untuk *autoencoder denoising* [10], [27] .

$$\mathcal{L}_{DAE}(X, X') = -\sum_{i=1}^{n} \log P(X_i \mid X') \tag{2.1}$$

dengan $\mathcal{L}_{DAE}(X, X')$ adalah fungsi *loss* BART, X urutan token asli, X' urutan token yang dimodifikasi, X_i urutan token ke-i dari X, dan $P(X_i \mid X')$ probabilitas prediksi token ke-i berdasarkan X'.



Gambar 2.2 Alur Kerja Model BART (digambar ulang oleh penulis)

2.4.1 *Tokenizer*

Tokenizer adalah komponen penting dalam model *seq2seq* yang mengonversi teks menjadi representasi numerik agar dapat diproses oleh model [28]. Dalam model BART, teknik *tokenizer* yang digunakan adalah metode *Byte-Pair Encoding* (BPE) [29]. Keunggulan utama BPE adalah kemampuannya mengatasi masalah kata-kata langka (*out-of-vocabulary words*) dengan efektif, serta fleksibilitasnya dalam menangani bahasa yang memiliki banyak kata majemuk atau variasi bentuk kata yang berubah sesuai konteks [29]. Hal ini membuat BPE sangat cocok untuk digunakan dalam NLP, terutama pada model seperti BART yang memerlukan tokenisasi yang efisien dan akurat [29]. Proses BPE diawali dengan memisahkan kata menjadi huruf atau karakter individu, kemudian setiap karakter tersebut dipasangkan dengan karakter yang mengikuti secara berurutan (misalnya “s” dengan “a”, “a” dengan “y”, “y” dengan “a”), dan frekuensi kemunculan setiap pasangan tersebut dihitung. Pasangan dengan frekuensi tertinggi kemudian digabungkan menjadi sebuah *subword* baru yang dimasukkan ke dalam kosakata model [29]. Berikut ini adalah langkah-langkah

lengkap dari proses BPE [29] :

1. Inisialisasi *vocabulary* :
   Setiap kata dipecah menjadi karakter atau huruf-huruf terpisah. Contohnya adalah kata "saya", ditulis menjadi (s a y a). Pada tahap awal, kosakata hanya berisi huruf-huruf tersebut.

2. Menghitung pasangan karakter :
   Setelah inisialisasi, dilakukan perhitungan frekuensi kemunculan setiap pasangan karakter yang berdekatan dalam seluruh teks, seperti (s,a), (a,y), dan (y,a).

3. Memilih pasangan karakter dengan frekuensi tertinggi :
   Dari hasil perhitungan tersebut, pasangan karakter dengan frekuensi kemunculan paling tinggi dipilih untuk digabungkan, seperti karakter "a" yang lebih banyak muncul setelah kata "s" dalam data.

4. Penggabungan pasangan karakter :
   Pasangan karakter dengan frekuensi tertinggi tersebut, digabungkan menjadi satu unit baru (misalnya (s,a) menjadi (sa)), kemudian seluruh kemunculannya dalam teks diganti dengan unit baru tersebut. Sebagai contoh, "s a y a" berubah menjadi "sa y a".

5. Pengulangan proses :
   Langkah 2 – 4 dilakukan berulang-ulang hingga tidak ada lagi pasangan huruf yang sering muncul. Proses ini menghasilkan daftar potongan kata baru (*subword*).

6. Penghentian proses :
   Jika tidak ada pasangan karakter yang bisa digabung, maka potongan kata (*subword*) yang ada dianggap sebagai hasil akhir.

7. Konversi ke representasi numerik :
   Setiap *subword* dipetakan ke dalam sebuah indeks numerik unik (ID), sehingga teks dapat direpresentasikan sebagai deret angka (*input_ids*) yang sesuai untuk diproses oleh model.

2.4.2 *Attention Mechanism*

Dalam arsitektur BART, terdapat metode *attention mechanism* yang diperkenalkan oleh Bahdanau dkk. (2015) dalam model *encoder - decoder*. *Encoder* mengubah *input* teks menjadi representasi vektor, sementara *decoder* menghasilkan *output* secara berurutan berdasarkan vektor tersebut [25]. Metode ini memungkinkan model memilih bagian *input* yang paling relevan, sehingga efektif dalam menerjemahkan kalimat panjang [30]. Langkah - langkah penyelesaiannya adalah sebagai berikut :

- *Attention Scores*
  Attention scores menentukan seberapa relevan *Query* (Q) dan *Key* (K) dalam mekanisme perhatian. Dalam konteks ini, *query* merepresentasikan kata yang sedang diproses dan *key* merepresentasikan kata yang sedang dibandingkan dengan *query*. Pada model *transformer*, metode yang dipakai adalah fungsi skor spesifik berupa *scaled dot-product*, yaitu metode menghitung *attention score* antara *query* dan *key* yang dirumuskan pada Persamaan (2.2) [31]:

$$e_{ij} = \frac{Q_i \cdot K_j}{\sqrt{d_k}} \tag{2.2}$$

dengan e_{ij} yaitu skor atensi antara *query* posisi-i dan *key* posisi-j, Q_i dan K_j adalah vektor *query* dan *key*, dan d_k yaitu dimensi vektor untuk normalisasi.

- *Attention Alignment*
  Setelah *attention scores* dihitung, dilakukan penyeimbangan (*alignment*) untuk mengubah skor tersebut menjadi bobot probabilistik dengan total sama dengan 1. Proses ini dapat dihitung melalui persamaan berikut [27]:

$$\alpha_l = \frac{\exp(e_l)}{\sum_{j=1}^{n} \exp(e_j)} \tag{2.3}$$

  dengan α_l adalah bobot perhatian pada posisi ke-l, e_l adalah *attention score*, dan $\sum_{j=1}^{n} \exp(e_j)$ adalah penjumlahan semua skor eksponensial.

- *Context Vector Calculation (Weighted Average)*
  Tahap akhir adalah menghitung *context vector* yang didapatkan melalui kombinasi linear dari *Value* (V) menggunakan bobot perhatian yang telah dihitung sebelumnya dengan persamaan berikut [27]:

$$c = \sum_{l=1}^{n} \alpha_l v_l \tag{2.4}$$

  dengan c adalah *context vector* hasil dari *attention mechanism*, α_l merupakan bobot perhatian, dan v_l adalah *value vector* ke-l.

2.4.3 *Fine-Tuning*

Fine-tuning adalah proses penyesuaian model setelah di *pre-trained* dengan dataset tambahan khusus [32]. Proses ini dapat meningkatkan akurasi model dalam memahami dan menghasilkan bahasa teknis pada suatu bidang tertentu [33]. *Fine-tuning* juga dapat dioptimalkan agar model cepat beradaptasi, sehingga akan meminimalkan kesalahan model dalam memproses data tersebut [34]. Selain itu, *fine-tuning* dapat mengurangi penggunaan memori sehingga memungkinkan model dijalankan pada perangkat dengan sumber daya terbatas [35].

2.5 Metode Evaluasi

Penelitian ini menggunakan *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE *score*) sebagai metode evaluasi utama untuk mengukur kesamaan antara ringkasan model dan ringkasan referensi melalui analisis *n-gram* (rangkaian n kata yang muncul secara berurutan dalam teks) serta kesesuaian urutan kata secara keseluruhan [36]. ROUGE *score* memberikan nilai numerik yang dapat digunakan untuk mengukur tingkat kesamaan antara ringkasan [36]. Metrik yang digunakan adalah ROUGE-N yang mengukur kesamaan ringkasan berdasarkan jumlah *n-gram* dan dapat dihitung untuk berbagai nilai n (misalnya ROUGE-1 untuk *unigram*, ROUGE-2 untuk *bigram*, ROUGE-3 untuk *trigram*, dan nilai n yang lebih besar). Pada penelitian ini digunakan ROUGE-1 dan ROUGE-2. Selain itu, digunakan juga ROUGE-L yang mengukur kesamaan ringkasan berdasarkan subsekuens kata terpanjang yang sama atau disebut juga *Longest Common Subsequence* (LCS), sehingga lebih cocok untuk mengevaluasi model peringkasan abstraktif [10]. Semakin tinggi nilai ROUGE, semakin besar kesamaan antara kedua ringkasan. Persamaan

ROUGE *score* dapat dihitung menggunakan rumus berikut [10]:

$$\text{ROUGE-N} = \frac{p}{q} \tag{2.5}$$

dengan ROUGE-N sebagai skor kecocokan *n-gram*, p adalah jumlah *n-gram* cocok antara ringkasan model dan referensi, serta q adalah total *n-gram* pada referensi.

$$\text{ROUGE-L} = \frac{\text{LCS}}{m} \tag{2.6}$$

dengan ROUGE-L sebagai skor berdasarkan subsekuens kata terpanjang sama (LCS), LCS panjang subsekuens yang cocok, dan m jumlah kata pada ringkasan referensi.

2.6 arXiv sebagai Sumber Data Peringkasan Teks

Sebagai repositori *preprint online* terkemuka, arXiv menyediakan akses terbuka ke berbagai bidang ilmu, seperti fisika, matematika, sains, dan ilmu komputer [37]. Platform ini memungkinkan peneliti untuk membagikan hasil riset mereka sebelum proses *peer-review* [37]. arXiv menyimpan hampir 2,5 juta artikel, sehingga menjadi salah satu dataset penelitian akademik terbesar yang tersedia untuk publik [38]. arXiv menawarkan berbagai fasilitas bagi peneliti, seperti mengunggah, memproduksi artikel, hingga menyediakan fitur pencarian, pengambilan data, dan distribusi melalui *website*. Selain itu, arXiv juga menyediakan akses API untuk sistem otomatis secara gratis dan dapat diakses secara langsung [5].

BAB III

Metode Penelitian

3.1 Deskripsi Data

Penelitian ini berfokus pada artikel ilmiah yang berkaitan dengan topik *"data science"*, seperti *machine learning, deep learning,* dan *data mining*. Data untuk penelitian ini dikumpulkan melalui proses *web scraping* menggunakan *arXiv Open Access API* yang memungkinkan akses langsung ke koleksi artikel prapublikasi secara terbuka. Proses *scraping* dilakukan dengan menyaring metadata artikel berdasarkan *query* tertentu terkait *data science*, rentang waktu publikasi, serta batas jumlah artikel yang diambil, seperti parameter yang terdapat pada Tabel 3.1. Sementara itu, struktur dan contoh dataset hasil *scraping* terdapat pada Tabel 3.2, yang mencakup informasi seperti judul, penulis, tanggal publikasi, *link file* PDF, dan abstrak untuk setiap artikel.

Tabel 3.1 Parameter *Scraping*

Parameter	**Nilai**
Query	*"data science"*
Start Date	2014-01-01
End Date	2024-12-31
Total Results	10000
Max Results Per Batch	100

3.2 Tahapan Pelaksanaan Penelitian

Penelitian ini diawali dengan pengumpulan data artikel ilmiah melalui arXiv API yang kemudian diproses pada tahap *preprocessing*. Proses ini mencakup ekstraksi teks dari *file* PDF,

Tabel 3.2 Dataset Hasil *Scraping*

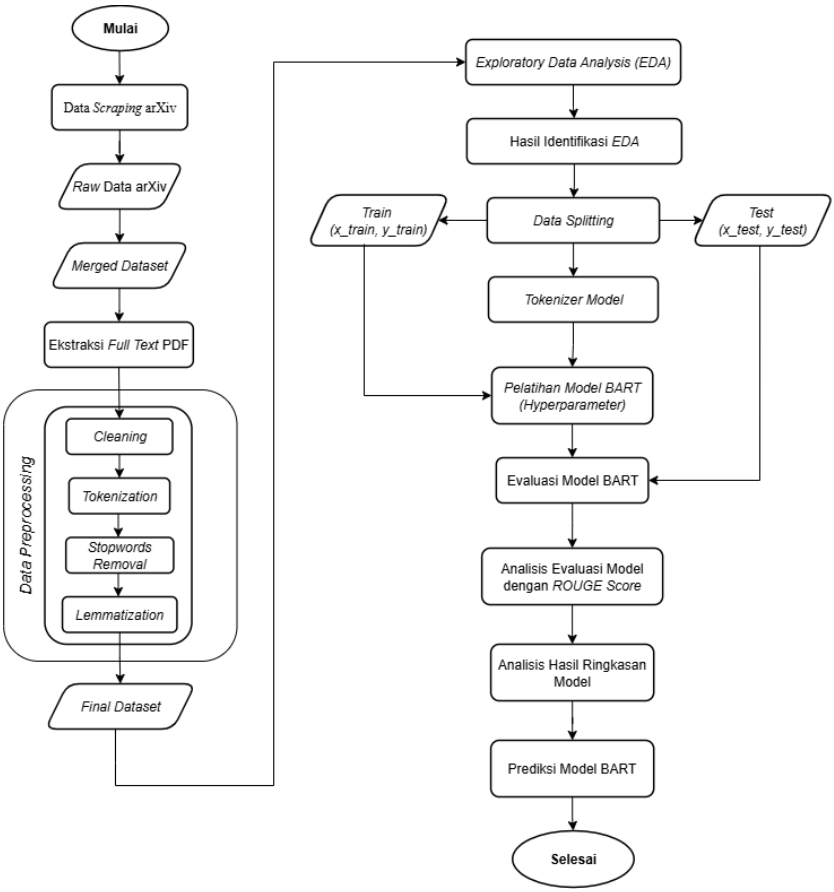
No	Title	Summary	Published	PDF Link	Authors
0	*Clinical Trial Information Extraction with BERT*	*Natural language processing (NLP) of clinical trial documents can be useful...*	2021-09-11	http://arxiv.org/pdf/2110.10027v1	*Xiong Liu, Greg L. Hersch, Iya Khalil, Murthy Devarakonda*
1	*Extraction of Medication and Temporal Relation from Clinical Text using Neural Language Models*	*Clinical texts, represented in electronic medical records (EMRs), contain...*	2023-10-03	http://arxiv.org/pdf/2310.02229v2	*Hangyu Tu, Lifeng Han, Goran Nenadic*
...	...	...	...	...	...
3839	*An AI-Assisted Design Method for Topology Optimization Without Pre-Optimized Training Data*	*Topology optimization is widely used by engineers during...*	2020-12-11	http://arxiv.org/pdf/2012.06384v2	*Alex Halle, L. Flavio Campanile, Alexander Hasse*

tokenisasi, *stopwords removal*, dan *lemmatization* untuk menyederhanakan teks sebelum pemodelan. *Input* data yang digunakan berupa teks hasil ekstraksi artikel ilmiah. Setelah proses

tersebut, dilakukan *exploratory data analysis* (EDA) pada data *scraping* dan data yang telah di *preprocessing* untuk memahami struktur data secara keseluruhan. Proses ini membantu dalam menganalisis karakteristik masing-masing data, serta mengevaluasi efektivitas tahap *preprocessing* terhadap peningkatan kualitas data. Selanjutnya, dilakukan pembagian data menjadi *data train* dan *data test* dengan rasio 80:20, dan juga dilakukan proses *tokenizer* untuk mengonversi teks menjadi representasi numerik. Model BART dilatih untuk menghasilkan ringkasan abstraktif dan evaluasinya dilakukan menggunakan ROUGE *score*. Visualisasi tahapan penelitian terdapat pada Gambar 3.1.

3.2.1 *Data Scraping*

Tahap pertama dalam penelitian ini adalah pengumpulan data artikel ilmiah menggunakan arXiv API yang bersifat publik dan tidak memerlukan *API key*. Pengambilan data berfokus pada artikel ilmiah dengan topik "*data science*", seperti *neural network, machine learning, AI*, dan *data mining*, dengan rentang waktu dari 1 Januari 2014 hingga 31 Desember 2024. Permintaan data dikirim secara bertahap dalam *batch* dengan maksimal 100 artikel per halaman, sesuai dengan batasan yang ditetapkan oleh ArXiv. Data yang diperoleh berbentuk XML (*Extensible Markup Language*), yaitu format berkas berbasis teks yang menyimpan data dalam struktur tag (menggunakan label) untuk menandai bagian-bagian informasi. Data ini diurai untuk mengekstrak informasi penting seperti judul, abstrak, tanggal publikasi, nama penulis, serta tautan *file* PDF artikel. Hasil ekstraksi tersebut disimpan dalam format *Comma-Separated Values* (CSV) untuk memudahkan proses analisis selanjutnya. Program fungsi yang digunakan pada tahap ini terdapat pada Lampiran B.



Gambar 3.1 Diagram Alur Penelitian

3.2.2 Ekstraksi *Full Text* PDF

Setelah tahap *scraping*, proses ini dimulai dengan mengunduh *file* PDF dari artikel ilmiah yang terdapat dalam dataset. Setiap artikel memiliki tautan PDF yang disediakan dalam kolom dataset, dan artikel-artikel ini diunduh secara otomatis menggunakan perintah *PdfReader* dari *library PyPDF2* [39]. Setelah diunduh, *file* disimpan dalam direktori lokal untuk memudahkan pengelolaan dan akses *offline*. Kemudian, dilakukan ekstraksi teks dari setiap

halaman artikel untuk digabung menjadi satu teks lengkap. Hasil ekstraksi ini disimpan dalam kolom baru pada dataset yang berisi konten artikel secara utuh. Namun, proses ekstraksi ini hanya mencakup konten berbasis teks. Elemen non-teks seperti tabel, rumus, dan gambar tidak diekstraksi secara langsung karena keterbatasan dari pustaka ekstraksi PDF yang digunakan. Dalam konteks ini, elemen-elemen tersebut dapat muncul sebagai teks tidak terstruktur, sehingga tidak digunakan dalam proses pemodelan.

3.2.3 *Data Preprocessing*

Pada tahap ini, dilakukan pembersihan data (*data cleaning*) dan persiapan data (*data preparation*). Pertama, data dibersihkan dengan menghapus kolom kosong dan karakter yang tidak sesuai seperti simbol atau tanda baca yang tidak memiliki makna dalam analisis teks [40]. Selanjutnya, dilakukan normalisasi teks seperti mengubah seluruh huruf menjadi huruf kecil menggunakan perintah *str.lower()*, menghilangkan angka, menghapus tanda baca seperti titik atau koma, serta mengurangi spasi berlebih [41]. Selain itu, dilakukan juga perbaikan ejaan kata menggunakan kamus *"frequency_dictionary_en_82_765.txt"*. Kamus ini memuat lebih dari 80 ribu entri kata dalam bahasa Inggris. Setiap entri dilengkapi dengan angka frekuensi yang menunjukkan seberapa sering kata tersebut muncul dalam kumpulan teks bahasa Inggris. Oleh karena itu, kamus ini tidak hanya menyediakan daftar kata yang baku, tetapi juga memberikan informasi seberapa sering atau jarang suatu kata digunakan. [42].

Setelah tahap pembersihan, teks diproses melalui *tokenization*, yaitu membagi teks menjadi token atau kata-kata menggunakan fungsi *word_tokenize* dari *library* NLTK untuk memudahkan model

memahami struktur kalimat dalam data [43]. Setelah proses *tokenization* selesai, dilakukan *stopwords removal*, di mana kata-kata dan istilah teknis tertentu yang tidak bermakna dihapus dari dataset [44]. Selain menggunakan daftar bawaan dari *library* NLTK, ditambahkan pula beberapa *stopwords* spesifik domain seperti yang direkomendasikan oleh Alshanik dkk. (2020) [44]. Tahap akhir adalah *lemmatization* yaitu mengubah kata menjadi bentuk dasarnya menggunakan *WordNetLemmatizer* dari *library* NLTK yang memanfaatkan *WordNet*, sebuah kumpulan kata dasar bahasa Inggris. Proses ini menghasilkan data teks yang terstandarisasi, bebas *noise*, dan siap digunakan, sehingga model dapat beroperasi secara efektif dalam menghasilkan ringkasan.

3.2.4 *Exploratory Data Analysis (EDA)*

Tahap eksplorasi data bertujuan untuk memahami karakteristik kata dan pola yang ada dalam dataset. Proses ini dilakukan pada 2 data, yaitu data mentah hasil *scraping* dan data yang telah di *preprocessing*. Tujuannya adalah untuk menganalisis karakteristik masing-masing data, serta mengevaluasi efektivitas tahap *preprocessing* terhadap peningkatan kualitas data. Pertama, semua data *scraping* dan data yang telah di *preprocessing* dihitung frekuensi kemunculannya menggunakan modul *Counter* untuk menampilkan token yang paling sering muncul dalam amsing-masing dataset. Selain itu, distribusi kata juga ditampilkan menggunakan *WordCloud*, di mana ukuran *font* akan mencerminkan frekuensi kemunculan setiap kata dalam dataset. Semakin besar ukuran kata tersebut, maka semakin sering kata tersebut muncul dalam data. Panjang kalimat dalam masing-masing dataset juga diukur berdasarkan jumlah kata per kalimat, dan divisualisasikan menggunakan *histogram* untuk mengidentifikasi pola distrbusinya.

3.2.5 *Data Splitting*

Data yang telah di *preprocessing*, dibagi menjadi data pelatihan (*data train*) dan data pengujian (*data test*) menggunakan fungsi *train_test_split*. Pembagian ini bertujuan untuk memastikan model tidak hanya mengingat data yang telah dipelajari dalam proses pelatihan, tetapi juga mampu melakukan generalisasi terhadap data baru yang belum pernah dilihat sebelumnya. Data *train* digunakan untuk melatih model BART agar mempelajari pola serta relasi antara teks artikel ilmiah dan ringkasannya. Sementara itu, data *test* digunakan untuk mengevaluasi kemampuan model dalam merangkum teks yang tidak ada di dalam data pelatihan. Selain itu, token-token hasil *preprocessing* digabungkan kembali menjadi kalimat menggunakan fungsi *' '.join(x)* [45]. Penggabungan ini bertujuan agar model dapat memahami informasi dari teks asli (*input*) secara lebih jelas, sehingga *output* yang dihasilkan sesuai dengan konteks teks aslinya. Pembagian data dilakukan dengan proporsi 80% untuk *training* dan 20% untuk *testing*.

3.2.6 *Tokenizer* dalam Model BART

Tahap selanjutnya adalah proses *tokenizer* yang mengubah data teks ke dalam format numerik menggunakan *BartTokenizer* dari *Hugging Face*. Setiap unit teks diurai menjadi *subword* token melalui algoritma *Byte-Pair Encoding* (BPE), kemudian dikonversi menjadi ID numerik unik berdasarkan *vocabulary* model. Model BART memiliki batasan panjang *input* yaitu maksimal 1024 token. Kemudian, sisa tokennya akan dilakukan pemotongan (*truncation*) yang mempertahankan segmen awal teks sebagai bagian paling informatif. Sebaliknya, teks yang lebih pendek dilengkapi dengan token *padding* ([PAD]) dan dilengkapi *attention mask* untuk membedakan teks asli dari *padding*. Hasil *tokenizer* ini berupa data

numerik yang kemudian di konversi menjadi *tensor PyTorch*. Proses ini berguna untuk menjaga informasi teks tetap utuh, sehingga model dapat memahami struktur dan konteksnya secara lebih baik saat dilatih maupun saat meringkas teks.

3.2.7 Pelatihan Model BART

Pelatihan model BART ini melibatkan dua komponen utama, yaitu mekanisme perhatian (*attention mechanism*) dan *fine-tuning*. *Attention mechanism* membantu model mengidentifikasi bagian input yang paling relevan untuk menghasilkan ringkasan informatif. Sementara itu, *fine-tuning* dilakukan untuk menyesuaikan model *pre-trained* agar dapat meningkatkan pemahaman model terhadap struktur dan gaya penulisan artikel ilmiah. *Fine-tuning* mencakup penyesuaian bobot model dan *hyperparameter* dengan konfigurasi awal seperti pada Tabel 3.3 [10]. *Learning rate* mengontrol seberapa besar perubahan bobot dalam setiap pembaruan selama pelatihan, *batch size* menentukan banyaknya sampel data yang diolah dalam satu iterasi, *epoch* menunjukkan berapa kali keseluruhan data dilatih, dan *weight decay* membantu menghindari *overfitting* dengan mengurangi besaran bobot model. Pemisahan *batch size* menjadi *batch size train* dan *test* dilakukan untuk mengoptimalkan penggunaan memori dan efisiensi model, proses pelatihan memerlukan *update* parameter sehingga lebih sensitif terhadap ukuran *batch*, sedangkan evaluasi tidak melibatkan proses gradien dengan ukuran *batch* yang lebih fleksibel, seperti yang dijelaskan dalam dokumentasi *Transformer* di *Hugging Face* [46]. Selanjutnya, beberapa *hyperparameter* divariasikan untuk mengetahui konfigurasi yang paling optimal pada data artikel ilmiah yang digunakan.

Tabel 3.3 Pengujian *Hyperparameter*

Parameter	Nilai
Learning Rate	0.0001
Batch Size Train	8
Batch Size Test	4
Number of Epochs	5
Weight Decay	0.01

3.3 Evaluasi Model

Setelah tahap pelatihan selesai, model kemudian diuji menggunakan data *test* untuk mengukur performanya dalam menghasilkan ringkasan abstraktif. Artikel-artikel ilmiah dari data *test* dimasukkan sebagai *input*, kemudian diolah oleh model BART untuk membuat ringkasan otomatis yang mencakup poin-poin penting berdasarkan pola bahasa dan struktur teks yang telah dipelajari selama pelatihan. Proses ini juga memanfaatkan teknik *sliding window* untuk mengatasi batasan panjang *input* model BART (1024 token), dengan cara memecah teks menjadi potongan lebih kecil (*chunks*) yang kemudian diringkas satu per satu menggunakan fungsi *bart_summarize()* dan digabung kembali untuk membentuk ringkasan akhir. Metode evaluasi yang digunakan adalah perhitungan ROUGE *Score*, yaitu metrik yang membandingkan kesamaan *n-gram* antara ringkasan model dan ringkasan referensi [47]. Data *test* diproses menggunakan *DataLoader* dan ringkasan yang dihasilkan model dibandingkan dengan ringkasan referensi untuk menghitung nilai ROUGE. Nilai ROUGE yang tinggi menunjukkan kualitas ringkasan yang lebih baik dan akurat, sehingga proses ini diharapkan dapat membantu peneliti dan pembaca memahami inti artikel ilmiah dengan lebih cepat. Contoh hasil ringkasan model ditunjukkan pada Tabel 4.13.

3.4 Prediksi Model

Setelah seluruh tahapan selesai, model diuji untuk menghasilkan ringkasan dari teks ilmiah yang belum pernah diproses sebelumnya, baik dalam proses pelatihan maupun pengujian. Pada tahap ini, model menerima *input* berupa teks artikel lengkap yang sudah di ekstrkasi dan tidak melalui tahap *preprocessing*. Batas *input* maksimal model BART yang hanya dapat memproses 1024 token diatasi dengan menggunakan metode *sliding window* yang membagi teks menjadi beberapa potongan (*chunks*) berukuran 512 token dengan *stride* sebesar 256 token agar konteks antar bagian tetap terjaga [46]. Setiap *chunk* kemudian dikonversi kembali menjadi teks dan diringkas menggunakan fungsi *bart_summarize()* dari *library* Hugging Face, sebelum digabungkan dan diringkas ulang menjadi satu ringkasan akhir yang utuh dan informatif. Berdasarkan pola bahasa dan informasi yang telah dipelajari selama pelatihan, model dapat mengekstrak poin-poin penting dari artikel baru tersebut sehingga pembaca, baik peneliti maupun mahasiswa, dapat memahami isi artikel secara cepat tanpa harus membaca keseluruhan dokumen secara mendetail. Proses ini diharapkan dapat meningkatkan efisiensi dalam meninjau dan memilih referensi ilmiah yang relevan dengan kebutuhan pembaca.

BAB IV

Hasil dan Pembahasan

4.1 Hasil Ekstraksi *Full Text* PDF

Data yang telah didapatkan dari proses *scraping* (pada Tabel 3.2), kemudian dilakukan ekstraksi untuk mendapatkan keseluruhan teks artikel ilmiah. Saat melakukan *scraping data*, didapatkan tautan untuk mengakses berkas setiap artikel, sehingga untuk mengetahui keseluruhan teks artikel tersebut dibutuhkan proses tambahan. Proses ini dilakukan dengan mengunduh berkas artikel ilmiah melalui tautan yang telah didapatkan, kemudian dilakukan ekstraksi teks untuk setiap dokumen artikel ilmiah. Setelah itu, didapatkan keseluruhan isi teks artikel ilmiah dengan lengkap (pada Tabel 4.1) yang disimpan dalam kolom *"full_text"* pada dataset. Proses ini hanya mencakup ekstraksi konten berbasis teks. Elemen non-teks seperti tabel, rumus, atau gambar tidak diekstraksi karena adanya keterbatasan dari *library* ekstraksi PDF yang digunakan. Implementasi lengkap dari fungsi program yang digunakan untuk proses ini terdapat pada Lampiran C.

Tabel 4.1 Dataset Hasil Ekstraksi Teks

PDF Link	***Full Text PDF***
http://arxiv.org/pdf/2110.10027v1	*Clinical trial designs are documented in unstructured text and natural language processing (NLP) of the documents...*
http://arxiv.org/pdf/2310.11046v2	*The prevalence of internet technology has accumulated a large amount of graph-structured data, which is widely used in Web applications...*

4.2 Hasil Data *Preprocessing*

Pada proses ini dilakukan tahap *cleaning* data untuk menghapus karakter-karakter yang tidak relevan, mengubah seluruh huruf kapital menjadi huruf kecil, dan penghapusan kolom kosong [48]. Setelah data di *cleaning*, dilakukan koreksi ejaan untuk memperbaiki kesalahan ejaan pada data. Selanjutnya, dilakukan tahap *preprocessing* seperti *tokenization* untuk mengubah kalimat menjadi kata-kata atau token, *stopwords removal* untuk menghapus kata-kata yang tidak bermakna, dan *lemmatization* untuk mengubah setiap kata menjadi bentuk kata baku atau kata dasarnya. Proses ini bertujuan untuk menghasilkan teks yang lebih bersih dan akurat, sehingga dapat membantu meningkatkan kualitas hasil ringkasan. Implementasi fungsi program untuk tahap ini dapat dilihat pada Lampiran D.

4.2.1 Hasil *Cleaning* Data

Pada tahap ini diterapkan proses *cleaning* untuk membersihkan data hasil ekstraksi dari karakter-karakter yang tidak relevan seperti angka, simbol, serta pembersihan kolom yang kosong atau NaN [48]. Tahap ini juga mencakup pengubahan seluruh huruf kapital menjadi huruf kecil untuk mempermudah analisis lebih lanjut dan menghindari perbedaan yang disebabkan oleh penggunaan huruf besar dan kecil (pada Tabel 4.2) [42]. Teks yang telah dibersihkan kemudian melalui proses koreksi ejaan, sehingga kata-kata yang salah tulis diperbaiki dan hasil teks menjadi lebih rapi seperti pada Tabel 4.3. Proses pembersihan dan koreksi ejaan ini menghasilkan teks yang lebih bersih dan akurat, kemudian disimpan dalam kolom baru pada dataset.

Pada Tabel 4.2, terlihat bahwa data telah di *cleaning* dengan baik. Pertama, seluruh teks telah diubah menjadi huruf kecil untuk

menyeragamkan dan menghilangkan variasi kata yang tidak diperlukan. Selanjutnya, tanda baca seperti titik, koma dan tanda kurung juga telah dibersihkan dari data karena tidak memberikan makna yang signifikan terhadap teks [48]. Proses ini juga mencakup perbaikan kata dengan ejaan yang salah, seperti kata "NIP" yang dikoreksi menjadi "nlp" (pada Tabel 4.3). Hasil *cleaning* ini membuat teks menjadi lebih rapih dan terstruktur.

Tabel 4.2 Dataset Hasil *Cleaning*

Full Text PDF	***Cleaned Full Text***	***Summary***	***Cleaned Summary***
Clinical trial designs are documented in unstructured text and natural language processing (NIP) of the documents arXiv.v...	*clinical trial designs are documented in unstructured text and natural language processing nip of the documents arxivv...*	*Natural language processing (NIP) of clinical trial documents can be useful in new trial design...*	*natural language processing nip of clinical trial documents can be useful in new trial design...*
The prevalence of internet technology has accumulated a large amount of graph-structured data, which is widely used in Web applications...	*the prevalence of internet technology has accumulated a large amount of graph structured data which is widely used in web applications...*	*The rapid development of Internet technology has given rise to a vast amount of graph-structured data...*	*the rapid development of internet technology has given rise to a vast amount of graph structured data...*

Tabel 4.3 Dataset Hasil Koreksi Ejaan

Cleaned Full Text	***Corrected Full Text***	***Cleaned Summary***	***Corrected Summary***
clinical trial designs are documented in unstructured text and natural language processing nip of the documents arxivv...	*clinical trial designs are documented in unstructured text and natural language processing nlp of the documents arxivv...*	*natural language processing nip of clinical trial documents can be useful in new trial design...*	*natural language processing nlp of clinical trial documents can be useful in new trial design...*
the prevalence of internet technology has accumulated a large amount of graph structured data which is widely used in web applications...	*the prevalence of internet technology has accumulated a large amount of graph structured data which is widely used in web applications...*	*the rapid development of Internet technology has given rise to a vast amount of graph structured data...*	*the rapid development of Internet technology has given rise to a vast amount of graph structured data...*

4.2.2 Hasil *Tokenization*

Data yang sudah melewati tahap *cleaning*, dilanjutkan dengan tahap *tokenization*. Setiap teks dalam kolom *"corrected_full_text"* dan *"corrected_summary"* dipecah menjadi token, dan hasilnya disimpan dalam kolom baru yang diberi nama *"tokens"* dan *"tokens_summary"* (Tabel 4.3). Setelah itu, dilakukan proses normalisasi token untuk menyeragamkan bentuk kata. Pada tahap ini, setiap token diperiksa dan dibersihkan dari karakter non-alfabet yang mungkin masih tersisa, seperti angka atau tanda baca yang ikut terbawa saat pemisahan kata [41].

Pada Tabel 4.4, setiap kalimat diubah menjadi daftar kata yang terpisah, ditandai dengan adanya tanda kutip dan tanda koma diantara kata-kata tersebut. Seperti pada penelitian Gaduh

Hartawan dkk. (2024) yang menyatakan bahwa *tokenization* merupakan langkah penting dalam *preprocessing* yang memecah teks menjadi unit-unit kecil agar lebih mudah diproses oleh model [10]. Tanpa *tokenization*, model akan kesulitan untuk memproses kata satu per satu dalam teks mentah, sedangkan dengan pemecahan menjadi token model dapat menganalisis dan mempelajari data secara lebih tepat. Proses ini juga membantu dalam meningkatkan efisiensi komputasi karena model tidak perlu lagi menangani teks dalam bentuk mentah yang kompleks.

Tabel 4.4 Dataset Hasil *Tokenization*

Corrected Full Text	***Tokens***	***Corrected Summary***	***Tokens Summary***
clinical trial designs are documented in unstructured text and natural language processing nlp of the documents arxivv...	*['clinical', 'trial', 'designs', 'are', 'documented', 'in', 'unstructured', 'text', 'and', 'natural', 'language', 'processing', 'nlp', 'of', 'the', 'documents', 'arxivv'...]*	*natural language processing nlp of clinical trial documents can be useful in new trial design...*	*['natural', 'language', 'processing', 'nlp', 'of', 'clinical', 'trial', 'documents', 'can', 'be', 'useful', 'in', 'new', 'trial', 'design'...]*
the prevalence of internet technology has accumulated a large amount of graph structured data which is widely used in web applications...	*['the', 'prevalence', 'of', 'internet', 'technology', 'has', 'accumulated', 'a', 'large', 'amount', 'of', 'graph', 'structured', 'data', 'which', 'is', 'widely', 'used', 'in', 'web', 'applications'...]*	*the rapid development of Internet technology has given rise to a vast amount of graph structured data...*	*['the', 'rapid', 'development', 'of', 'internet', 'technology', 'has', 'given', 'rise', 'to', 'a', 'vast', 'amount', 'of', 'graph', 'structured', 'data'...]*

4.2.3 Hasil *Stopwords Removal*

Pada Tabel 4.5, kata-kata yang tidak memberikan makna khusus pada teks dihapus, seperti kata "arxivv" dan *template* kata seperti "*ARTICLE TEMPLATE*" [44]. Namun, tidak semua kata umum seperti *"of", "the", "and"*, dan *"in"* dihilangkan dari teks, karena model BART memerlukan struktur kalimat yang utuh untuk memahami konteks teks secara menyeluruh. Oleh karena itu, *stopwords removal* hanya diterapkan pada kata-kata yang benar-benar tidak bermakna atau kata-kata dengan urutan tidak jelas, agar hubungan antar kata dalam teks tetap terjaga. Proses ini membantu mengurangi kata-kata yang tidak relevan sehingga teks menjadi lebih fokus pada konten utama artikel. Hal ini memungkinkan model BART mengenali kata kunci penting secara lebih efektif, sehingga mendukung peningkatan kualitas ringkasan yang dihasilkan.

4.2.4 Hasil *Lemmatization*

Tahap akhir dari *preprocessing* adalah *lemmatization*, yaitu mengubah setiap kata menjadi bentuk dasarnya (lemma), seperti yang terlihat pada Tabel 4.6 [49]. Tahap ini mengubah kata-kata dengan variasi bentuk yang berbeda tetapi memiliki makna yang serupa. *Lemmatization* tidak hanya mengurangi pengulangan kata dalam teks, tetapi juga membantu membuat isi dokumen menjadi lebih jelas dan mudah dipahami. Proses ini diharapkan dapat menghasilkan representasi teks yang lebih konsisten dan bermakna, sehingga siap digunakan untuk analisis lebih lanjut.

Pada Tabel 4.6, setiap kata diubah menjadi bentuk dasarnya. Contohnya adalah kata "*designs*" menjadi "*design*", "*documented*" menjadi "*document*", dan "*given*" menjadi "*give*". Proses ini menyeragamkan variasi kata yang bermakna sama tetapi berbeda

Tabel 4.5 Dataset Hasil *Stopwords Removal*

Tokens	***Token Stopwords***	***Tokens Summary***	***Tokens Summary Stopwords***
['clinical', 'trial', 'designs', 'are', 'documented', 'in', 'unstructured', 'text', 'and', 'natural', 'language', 'processing', 'nlp', 'of', 'the', 'documents', 'arxivv'...]	*['clinical', 'trial', 'designs', 'are', 'documented', 'in', 'unstructured', 'text', 'and', 'natural', 'language', 'processing', 'nlp', 'of', 'the', 'documents'...]*	*['natural', 'language', 'processing', 'nlp', 'of', 'clinical', 'trial', 'documents', 'can', 'be', 'useful', 'in', 'new', 'trial', 'design'...]*	*['natural', 'language', 'processing', 'nlp', 'of', 'clinical', 'trial', 'documents', 'can', 'be', 'useful', 'in', 'new', 'trial', 'design'...]*
['the', 'prevalence', 'of', 'internet', 'technology', 'has', 'accumulated', 'a', 'large', 'amount', 'of', 'graph', 'structured', 'data', 'which', 'is', 'widely', 'used', 'in', 'web', 'applications'...]	*['the', 'prevalence', 'of', 'internet', 'technology', 'has', 'accumulated', 'a', 'large', 'amount', 'of', 'graph', 'structured', 'data', 'which', 'is', 'widely', 'used', 'in', 'web', 'applications'...]*	*['the', 'rapid', 'development', 'of', 'internet', 'technology', 'has', 'given', 'rise', 'to', 'a', 'vast', 'amount', 'of', 'graph', 'structured', 'data'...]*	*['the', 'rapid', 'development', 'of', 'internet', 'technology', 'has', 'given', 'rise', 'to', 'a', 'vast', 'amount', 'of', 'graph', 'structured', 'data'...]*

bentuk, sehingga dapat mengurangi terjadinya pengulangan kata. Hasil dari proses ini membantu model agar dapat mengenali kata-kata yang berbeda bentuk tetapi memiliki makna yang sama, sehingga tidak dianggap hal yang berbeda dalam analisis [49].

Tabel 4.6 Dataset Hasil *Lemmatization*

Token Stopwords	***Token Lemmatized***	***Tokens Summary Stopwords***	***Tokens Summary Lemmatized***
['clinical', 'trial', 'designs', 'are', 'documented', 'in', 'unstructured', 'text', 'and', 'natural', 'language', 'processing', 'nlp', 'of', 'the', 'documents'...]	*['clinical', 'trial', 'design', 'be', 'document', 'in', 'unstructured', 'text', 'and', 'natural', 'language', 'processing', 'nlp', 'of', 'the', 'document'...]*	*['natural', 'language', 'processing', 'nlp', 'of', 'clinical', 'trial', 'documents', 'can', 'be', 'useful', 'in', 'new', 'trial', 'design'...]*	*['natural', 'language', 'processing', 'nlp', 'of', 'clinical', 'trial', 'document', 'can', 'be', 'useful', 'in', 'new', 'trial', 'design'...]*
['the', 'prevalence', 'of', 'internet', 'technology', 'has', 'accumulated', 'a', 'large', 'amount', 'of', 'graph', 'structured', 'data', 'which', 'is', 'widely', 'used', 'in', 'web', 'applications'...]	*['the', 'prevalence', 'of', 'internet', 'technology', 'have', 'accumulate', 'a', 'large', 'amount', 'of', 'graph', 'structured', 'data', 'which', 'be', 'widely', 'use', 'in', 'web', 'application'...]*	*['the', 'rapid', 'development', 'of', 'internet', 'technology', 'has', 'given', 'rise', 'to', 'a', 'vast', 'amount', 'of', 'graph', 'structured', 'data'...]*	*['the', 'rapid', 'development', 'of', 'internet', 'technology', 'have', 'give', 'rise', 'to', 'a', 'vast', 'amount', 'of', 'graph', 'structured', 'data'...]*

4.3 Hasil *Exploratory Data Analysis* (EDA)

Exploratory Data Analysis (EDA) dilakukan pada dua versi dataset, yaitu data mentah hasil *scraping* dan data yang telah melalui *preprocessing*, untuk membandingkan karakteristik awal dan dampak *preprocessing* terhadap kualitas data. Analisis ini mencakup distribusi frekuensi kata, visualisasi *word cloud*, dan distribusi panjang kalimat. Hasilnya menunjukkan bahwa data

setelah *preprocessing* memiliki sebaran yang lebih beragam dibandingkan data mentah, sehingga membuktikan bahwa tahap *preprocessing* tidak hanya membersihkan, tetapi juga menghasilkan distribusi kalimat yang lebih alami [50]. Aspek ini penting dalam pelatihan model BART karena distribusi yang baik mendukung performa model secara optimal.

4.3.1 Hasil EDA Data *Scraping*

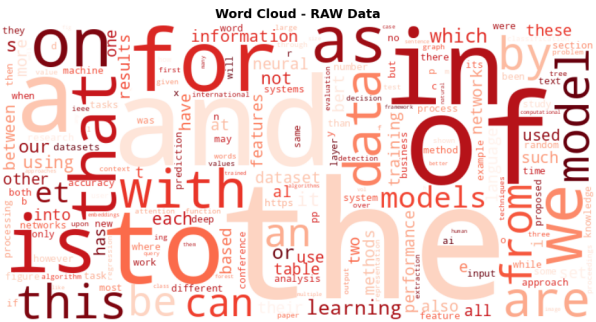
Tahap pertama dari proses ini adalah dilakukan perhitungan frekuensi kata untuk memahami distribusi kata yang paling sering muncul dalam data *scraping* (pada Tabel 4.7). Analisis ini membantu mengidentifikasi istilah umum yang paling sering digunakan dan paling jarang digunakan [50]. Selain itu, dilakukan juga visualisasi *word cloud* untuk memberikan gambaran mengenai kata-kata yang paling sering muncul dengan lebih menarik dan mudah dipahami. Semakin besar ukuran sebuah kata pada *Word Cloud*, artinya kata tersebut semakin sering muncul dalam teks [50]. Seperti pada Gambar 4.1, kata "*the, of, and, to, in*" memiliki ukuran yang lebih besar dibandingkan kata-kata lainnya. Tanda baca seperti tanda titik, tanda kutip, dan tanda kurung juga memiliki ukuran lebih besar dibandingkan yang lain dalam visualisasi tersebut. Hal ini menandakan bahwa kata dan tanda tersebut paling sering muncul dalam teks, sehingga memiliki ukuran yang lebih besar.

Proses ini juga mencakup analisis distribusi panjang kalimat dengan menghitung jumlah kata dalam setiap kalimat. Distribusi ini divisualisasikan dalam bentuk histogram untuk melihat sebaran panjang kalimat dalam dataset seperti yang ditampilkan pada Gambar 4.2. Analisis ini membantu mengidentifikasi dokumen yang sangat pendek atau sangat panjang, dan memberikan

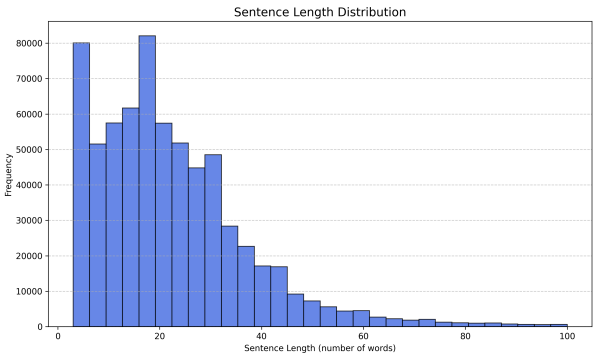
gambaran tentang variasi panjang dokumen dalam dataset [50]. Berdasarkan gambar tersebut mayoritas kalimat memiliki panjang antara 5-30 kata, dengan distribusi terbanyak di antara 10-20 kata. Rata-rata panjang kalimat dalam dataset adalah sekitar 18-20 kata.

Tabel 4.7 Sampel Distribusi Frekuensi Kata Data *Scraping*

Word	***Count***
the	659563
of	350102
and	337086
to	233520
in	231450



Gambar 4.1 *Word Cloud Data Scraping*



Gambar 4.2 Distribusi Panjang Kalimat Data *Scraping*

4.3.2 Hasil EDA Data *Preprocessing*

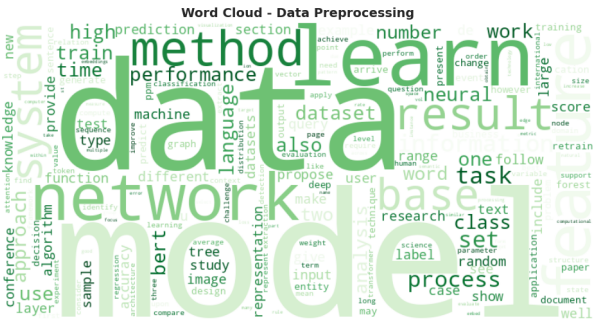
Proses ini dilakukan pada data yang telah melalui *preprocessing* dengan tahapan serupa seperti data *scraping*. Frekuensi kata dihitung untuk mengidentifikasi kata dominan dalam teks, seperti pada Tabel 4.8. Visualisasi *word cloud* juga dibuat menggunakan skema warna hijau (*Greens*), di mana kata yang sering muncul memiliki ukuran lebih besar, misalnya "*network, data, model, method, learn*" (pada Gambar 4.3) [50]. Selanjutnya, panjang kalimat dianalisis dengan menghitung jumlah token dalam setiap kalimat, lalu divisualisasikan dalam *histogram* (pada Gambar 4.4). Grafik tersebut menunjukkan sebagian besar kalimat memiliki panjang 5–40 kata, namun terdapat juga kalimat yang mencapai hampir 100 kata. Hal ini menggambarkan adanya keragaman struktur kalimat dalam dataset, mulai dari kalimat singkat yang padat informasi hingga kalimat yang panjang [50].

Jika dibandingkan dengan analisis data *scraping*, hasil setelah melalui tahap *preprocessing* menunjukkan efektivitas yang lebih tinggi dalam menyajikan informasi yang lebih bersih dan relevan. Pada data hasil *scraping*, kata-kata umum seperti *"the, of, and, to, in"* serta tanda baca mendominasi visualisasi *word cloud*, sehingga informasi yang diperoleh menjadi kurang bermakna. Namun, setelah *preprocessing* dilakukan, analisis frekuensi kata menjadi lebih terfokus pada istilah-istilah yang memiliki makna semantik penting, seperti *network, data, model, method, learn,"* sebagaimana terlihat pada Gambar 4.3. Visualisasi *word cloud* juga menjadi lebih representatif terhadap topik yang dibahas dalam teks. Selain itu, distribusi panjang kalimat juga menjadi lebih merata dan mencerminkan struktur kalimat yang lebih bersih, berbeda dengan distribusi sebelumnya yang masih dipengaruhi oleh *noise* dari teks mentah. Oleh karena itu, tahap *preprocessing* efektif dalam

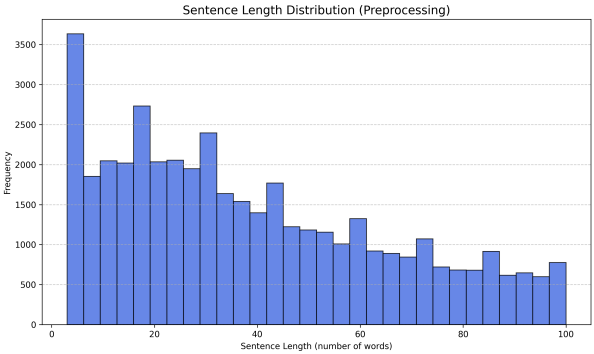
membantu meningkatkan kualitas analisis dan interpretasi data teks [10].

Tabel 4.8 Sampel Distribusi Frekuensi Kata Data *Preprocessing*

Word	***Count***
model	102676
data	74534
learn	36822
network	29660
method	28285



Gambar 4.3 *Word Cloud Data Preprocessing*



Gambar 4.4 Distribusi Panjang Kalimat Data *Preprocessing*

4.4 Hasil Data *Splitting*

Pada penelitian ini dataset yang berjumlah 2.318 dokumen dibagi menjadi dua bagian, dengan 80% sebagai data latih dan 20% sebagai data uji, seperti yang ditunjukkan pada Tabel 4.9. Pembagian ini dilakukan untuk memberikan cukup data bagi model dalam proses pelatihan, sementara *data test* digunakan untuk evaluasi akhir pada data yang belum pernah dilihat sebelumnya. Proses ini bertujuan untuk menguji performa model selama pelatihan, mencegah *overfitting*, dan memastikan kemampuan generalisasi model tetap baik [10]. Melalui pembagian ini, performa model dapat diukur secara lebih objektif, sehingga risiko *overfitting* dapat diminimalkan.

Tabel 4.9 Hasil Data *Splitting*

Training Set	***Test Set***
80% dari *Full Set* 1.854 Dokumen	20% dari *Full Set* 464 Dokumen

4.5 Hasil Tokenizer Model BART

Proses selanjutnya adalah memuat model dan tokenizer BART yang telah dilatih sebelumnya, yaitu *"facebook/bart-large-cnn"* yang secara khusus dirancang untuk tugas *text summarization*, seperti yang digunakan dalam penelitian oleh Gaduh Hartawan dkk. (2024) [10]. Model ini bertugas menghasilkan ringkasan dari teks panjang dengan tetap mempertahankan informasi-informasi penting di dalamnya. Selanjutnya, teks ringkasan diproses menggunakan metode *Byte Pair Encoding (BPE)*, dan kemudian dikonversi menjadi bentuk tensor. Agar *input* memiliki panjang yang seragam sesuai kebutuhan model, diterapkan *padding*. Setelah memastikan kesesuaian antara *input* dan label, data pelatihan dan pengujian

disusun dalam bentuk dataset yang siap digunakan oleh model. Terakhir, dilakukan pengecekan untuk memastikan struktur data telah sesuai untuk proses pelatihan.

4.6 Hasil Pelatihan Model BART

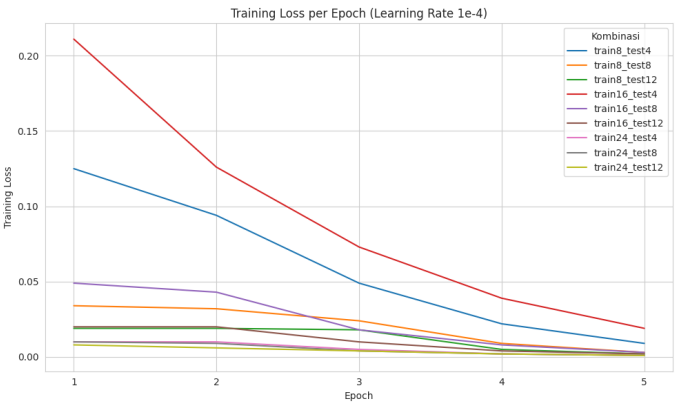
Konfigurasi pelatihan dilakukan menggunakan *TrainingArguments*, dengan berbagai parameter seperti *learning rate*, *batch size*, dan jumlah *epoch*. Proses evaluasi dilakukan setiap 1000 langkah untuk meningkatkan efisiensi penggunaan sumber daya. Namun, pengaturan *hyperparameter* yang digunakan sebelumnya (pada Tabel 3.3) belum memberikan hasil yang cukup baik seperti yang ditampilkan pada Tabel 4.11, sehingga dilakukan penyesuaian *hyperparameter* lebih lanjut untuk meningkatkan efektivitas model [10]. Selanjutnya, rentang *hyperparameter* seperti *learning rate* dan *batch size* diatur untuk mencapai hasil yang optimal sekaligus mempertimbangkan efisiensi sumber daya seperti pada Tabel 4.10. Proses pelatihan model BART dilakukan dengan memanfaatkan fungsi *Trainer* dari *library Transformers*.

Berdasarkan kombinasi tersebut, didapatkan nilai *training loss* untuk setiap kombinasi *hyperparameter*. Nilai *loss* ini bertujuan untuk mengetahui seberapa besar kesalahan model terhadap data yang dilatih. Nilai *loss* yang rendah dapat mengindikasikan bahwa model telah mempelajari pola data latih secara efektif [51]. Nilai *loss* yang dihasilkan pada kombinasi ini terdapat pada Gambar 4.5 dan 4.6. Berdasarkan grafik visualisasi tersebut, terlihat bahwa sebagian besar kombinasi dengan *learning rate* 1e-4 menghasilkan *loss* akhir yang relatif rendah, dengan rata-rata di bawah angka 0.05. Hal ini menunjukkan bahwa model berhasil belajar secara efektif dan stabil. Sementara itu, kombinasi dengan *learning rate* 1e-5 cenderung menghasilkan *training loss* yang lebih bervariasi,

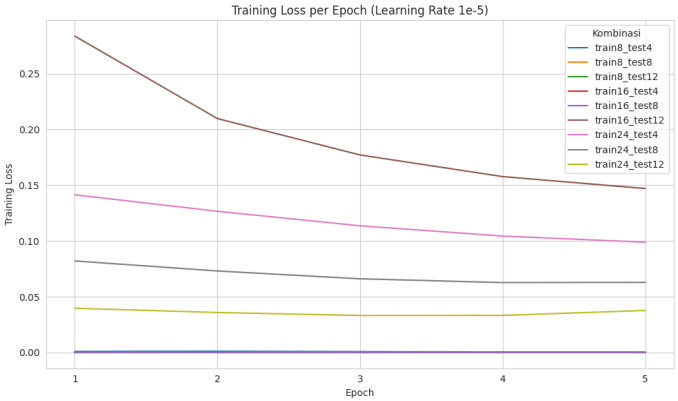
dengan rata-rata berkisar antara 0.00 hingga 0.28. Kondisi ini menunjukkan bahwa model belum belajar secara optimal yang disebabkan karena pembaruan bobot terlalu kecil atau memerlukan waktu lebih lama untuk menangkap pola data dengan efektif. Pola distribusi nilai *loss* dari berbagai kombinasi juga menunjukkan bahwa pada *epoch* ke-5, sebagian besar model telah mencapai titik stabil yang ditandai dengan tidak adanya nilai *loss* yang ekstrem atau menurun dengan tajam. Oleh karena itu, penggunaan 5 *epoch* sudah cukup untuk mencapai kestabilan model. Fungsi program *compute_metrics* yang digunakan untuk mengevaluasi performa model selama proses pelatihan ditunjukkan pada Lampiran E.

Tabel 4.10 Rentang Pengujian *Hyperparameter*

Parameter	**Nilai**
Learning Rate	0.0001, 0.00001
Batch Size Train	8, 16, 24
Batch Size Test	4, 8, 12
Number of Epochs	5
Weight Decay	0.01



Gambar 4.5 Grafik *Training Loss* untuk *Learning Rate* 1e-4



Gambar 4.6 Grafik *Training Loss* untuk *Learning Rate* 1e-5

4.7 Hasil Evaluasi Model BART

Tahap evaluasi model dilakukan menggunakan *test dataset* untuk menilai kinerja BART setelah proses pelatihan. Tujuan evaluasi ini adalah memastikan model mampu menghasilkan ringkasan yang sesuai dengan ringkasan referensinya. Penilaian dilakukan dengan metrik ROUGE (ROUGE-1, ROUGE-2, ROUGE-L) yang mengukur tingkat kesamaan antara ringkasan oleh model dan ringkasan asli [10]. Hasil evaluasi ini memberikan gambaran kuantitatif terhadap performa model BART dalam meringkas artikel ilmiah. Implementasi program ditampilkan pada Lampiran F.

4.7.1 Analisis ROUGE *Score*

Tahap evaluasi dilakukan dengan menghitung kesamaan antara ringkasan referensi dan ringkasan yang dihasilkan model BART menggunakan ROUGE *Score* [36]. Perhitungan skor ROUGE untuk setiap kombinasi hyperparameter (pada Tabel 4.10) menunjukkan bahwa konfigurasi terbaik terdapat pada *learning rate* 0.00001 (1e-5), *batch size train* 24, *batch size test* 4, *epoch* 5, dan *weight decay* 0.01, dengan skor ROUGE-1: 0.60, ROUGE-2: 0.52, dan

ROUGE-L: 0.56. Skor ini menunjukkan ringkasan model cukup baik dalam memahami ringkasan referensi berdasarkan *unigram, bigram*, dan urutan kalimat. Konfigurasi ini terbukti lebih unggul dibanding kombinasi lainnya, di mana *learning rate* 1e-5 menjadi faktor paling berpengaruh terhadap peningkatan skor ROUGE. Hal ini sejalan dengan penelitian oleh Zhao dan Chen (2022), yang menunjukkan bahwa *learning rate* 1e-5 memberikan hasil yang optimal dalam proses *fine-tuning* model BART untuk tugas peringkasan teks ilmiah [52]. Penelitian lain oleh Ulker dan Ozer (2024) juga mendukung temuan tersebut, dengan hasil yang menunjukkan bahwa *learning rate* serupa meningkatkan kualitas ringkasan yang dihasilkan oleh model BART [53].

Penggunaan *batch size train* sebesar 24 dan *batch size test* sebesar 4 dinilai cukup untuk menjaga kestabilan pelatihan sekaligus mempertimbangkan kapasitas memori. Pengaturan epoch sebesar 5 juga cukup untuk memastikan bahwa model mencapai konvergensi tanpa berlebihan sehingga menghindari *overfitting* pada data pelatihan. Seperti pada penelitian Popel dan Bojar (2018) yang menyatakan bahwa pelatihan model yang berlangsung terlalu lama (dalam hal jumlah langkah pelatihan) dapat menyebabkan *overfitting* dan penurunan kualitas pada data pengujian [54]. Penggunaan *weight decay* sebesar 0.01 berperan sebagai teknik regularisasi (*L2 regularization*), yaitu metode yang menambahkan penalti terhadap bobot bernilai besar pada fungsi *loss*. Pendekatan ini berbeda dengan *L1 regularization* yang cenderung menghasilkan bobot nol (*sparse*) maupun *dropout* yang menonaktifkan neuron secara acak. Dibandingkan keduanya, *L2 regularization* lebih konsisten dalam menjaga bobot tetap kecil dan terdistribusi merata, sehingga proses optimasi menjadi lebih stabil dan nilai *loss* lebih terkendali [55]. Oleh karena itu, konfigurasi *hyperparameter* yang

telah dipilih dapat dijadikan sebagai dasar untuk pengembangan model *summarization* yang lebih optimal.

Tabel 4.11 Evaluasi ROUGE *Score Hyperparameter* Rujukan

ROUGE-1	ROUGE-2	ROUGE-L
0.51	0.40	0.43

Tabel 4.12 Evaluasi ROUGE *Score* Kombinasi *Hyperparameter*

No.	*Learning Rate*	*Batch Size Train*	*Batch Size Test*	*Epoch*	*Weight Decay*	ROUGE-1	ROUGE-2	ROUGE-L
1	0.0001	8	4	5	0.01	0.53	0.40	0.46
2	0.0001	8	8	5	0.01	0.53	0.40	0.46
3	0.0001	8	12	5	0.01	0.52	0.40	0.46
4	0.0001	16	4	5	0.01	0.49	0.36	0.41
5	0.0001	16	8	5	0.01	0.52	0.40	0.46
6	0.0001	16	12	5	0.01	0.52	0.39	0.45
7	0.0001	24	4	5	0.01	0.49	0.35	0.41
8	0.0001	24	8	5	0.01	0.52	0.39	0.45
9	0.0001	24	12	5	0.01	0.51	0.38	0.44
10	0.00001	8	4	5	0.01	0.58	0.49	0.54
11	0.00001	8	8	5	0.01	0.58	0.49	0.54
12	0.00001	8	12	5	0.01	0.56	0.46	0.51
13	0.00001	16	4	5	0.01	0.59	0.51	0.55
14	0.00001	16	8	5	0.01	0.59	0.51	0.55
15	0.00001	16	12	5	0.01	0.59	0.51	0.55
16	0.00001	24	4	5	0.01	0.60	0.52	0.56
17	0.00001	24	8	5	0.01	0.60	0.51	0.56
18	0.00001	24	12	5	0.01	0.60	0.51	0.56

4.7.2 Analisis Hasil Ringkasan Model

Dari 20% data *test* yang terdiri dari 464 dokumen, diambil dua artikel ilmiah sebagai contoh evaluasi, yaitu *Clinical Trial Information Extraction with BERT* dan *FinBERT-MRC: Financial Named Entity Recognition using BERT under the Machine Reading Comprehension Paradigm*. Contoh ini ditampilkan dalam Tabel 4.13 dan untuk menilai keakuratan dan relevansi ringkasan yang

dihasilkan model terhadap teks aslinya. Hasilnya menunjukkan bahwa model mampu memahami tema dan struktur dasar teks ilmiah dengan cukup baik, meskipun ringkasan yang dihasilkan cenderung bersifat umum. Misalnya, pada artikel pertama ringkasan asli menjelaskan tantangan pengolahan *big data* dan memperkenalkan *library* Python bernama DataSist, sedangkan ringkasan model hanya menyebutkan analisis *big data* dan alat populer seperti Pandas dan SPSS, tanpa menyoroti pengembangan *library* tersebut. Hal ini menunjukkan bahwa model BART cenderung menghasilkan ringkasan yang bersifat deskriptif [25].

Di artikel kedua, ringkasan yang dihasilkan oleh model menunjukkan performa yang lebih baik. Model berhasil mengidentifikasi struktur dasar BiFlaG, termasuk dua modul utama serta teknik yang diterapkan seperti BiLSTM dan GCN. Namun, ringkasan tersebut masih melewatkan beberapa detail penting, seperti keunggulan model dibandingkan metode sebelumnya dan hasil eksperimen yang mendukung klaim tersebut. Selain itu, terdapat kekurangan dalam kelancaran alur kalimat dan pemilihan kosakata, sehingga narasi terlihat kurang alami dan belum sepenuhnya mencerminkan struktur teks asli. Penelitian Zhang dkk. (2020) juga menyebutkan bahwa model peringkasan seperti BART cukup handal dalam menangkap makna keseluruhan teks, tetapi masih memiliki kendala dalam menyampaikan informasi teknis atau data numerik secara akurat [56]. Hal ini mengindikasikan bahwa meskipun BART mampu memahami konteks umum dan topik utama artikel ilmiah, hasil ringkasan masih bisa ditingkatkan misalnya dengan menggabungkan pendekatan ekstraktif dan abstraktif.

Tabel 4.13 Hasil Ringkasan Model dengan Ringkasan Asli Artikel Ilmiah

Judul Artikel	**Teks Referensi**	**Teks Ringkasan BART**
Clinical Trial Information Extraction with BERT	*a large amount of data is produced every second from modern information systems such as mobile devices the world wide web internet of things social media etc analysis and mining of this massive data requires a lot of advanced tools and techniques therefore big data analytic and mining is currently an active and trending area of research because of the enormous benefits businesses and organizations derive from it numerous tools like pandas bumpy STATA, SPSS, have been created to help analyse and mine these huge outburst of data and some have become so popular and widely used in the field this paper presents a new pythonbased library DataSist, which offers high level intuitive and easy to use functions and methods that helps data scientistsanalyst to quickly analyses mine and visualize big data sets the objectives of this project were to i design a python library to aid data analysis process by abstracting low level syntax ii increase productivity of data scientist by making them focus on what to do rather than how to do it this project shows that data analysis can be automated and much faster when we abstract certain functions and will serve as an important tool in the workflow of data scientists.*	*Big data analytic and mining is currently an active and trending area of research because of the enormous benefits businesses and organizations derive from it numerous tools like pandas bumpy stata, spss, have been created to help analyse and mine these huge outburst of data big data as noted in a has been coined to represent this outburst of massive data that cannot fit into traditional database management tools or data processing applications these data are available in three different formats such as structured semistructured and unstructured and the sizes are in scales of terabytes and petabytes normally big data is categorized into dimensions.*

Tabel 4.13 (Lanjutan dari halaman sebelumnya)

Judul Artikel	Teks Referensi	Teks Ringkasan BART
FinBERT-MRC: financial named entity recognition using BERT under the machine reading comprehension paradigm	*in this paper we propose a novel bipartite flatgraph network BiFlaG for nested named entity recognition NER, which contains two subgraph modules a flat NER module for outermost entities and a graph module for all the entities located in inner layers bidirectional LSTM BiLSTM and graph convolution network GCN are adopted to jointly learn flat entities and their inner dependencies different from previous models which only consider the unidirectional delivery of information from innermost layers to outer ones or outsidetoinside, our model effectively captures the bidirectional interaction between them be first use the entities recognized by the flat NER module to construct an entity graph which is fed to the next graph module the richer representation learned from graph module carries the dependencies of inner entities and can be exploited to improve outermost entity predictions experimental results on three standard nested NER datasets demonstrate that our BiFlaG outperforms previous stateoftheart models.*	*In this paper we propose a novel bipartite at graph network biflag for nested named entity recognition ner, which contains two subgraph modules a at ner module for out most entities and a graph module for all the entities located in inner layers bidirectional lstm bilstm and graph convolution net work gcn are adopted to jointly learn at entities and their inner dependencies differ significantly from previous models which only consider the unidirectional delivery of information from named entity recognition ner aims to identify words or phrases that contain the names of different categories like location organization or medical codes.*

4.8 Hasil Prediksi Model BART

Tahap ini menyajikan hasil prediksi model BART pada artikel ilmiah baru yang tidak termasuk dalam data latih maupun data uji.

Tujuan prediksi ini adalah untuk menilai sejauh mana model dapat beradaptasi dan menghasilkan ringkasan yang informatif serta relevan dari dokumen yang sama sekali belum pernah dilihat. Hal ini penting untuk mengetahui kemampuan model dalam memahami teks di luar data yang digunakan selama pelatihan dan pengujian. Ringkasan yang dihasilkan model kemudian dibandingkan dengan teks asli artikel ilmiah untuk mengevaluasi efektivitas BART dalam menyajikan poin-poin utama secara ringkas. Analisis ini memberikan tinjauan awal mengenai kemampuan BART dalam membantu pembaca menangkap inti artikel ilmiah tanpa perlu membaca keseluruhan teks. Implementasi fungsi program untuk tahap ini tercantum pada Lampiran F.

4.8.1 Hasil ROUGE *Score* Prediksi Model

Hasil pengujian menunjukkan bahwa model berhasil menghasilkan ringkasan dengan skor ROUGE sebesar 0.48 untuk ROUGE-1, 0.36 untuk ROUGE-2, dan 0.43 untuk ROUGE-L. Meskipun skor tersebut menunjukkan bahwa ringkasan yang dihasilkan masih cukup relevan dengan ringkasan referensi, nilainya cenderung lebih rendah dibandingkan hasil evaluasi pada data uji. Penurunan performa ini dapat disebabkan oleh beberapa faktor. Pertama, data prediksi berasal dari artikel yang memiliki distribusi berbeda atau berada di luar distribusi (*out-of-distribution*) data pelatihan, sehingga model kesulitan mengenali pola yang serupa, seperti yang dijelaskan dalam penelitian Danqing Wang dkk. (2019) [57]. Penelitian tersebut menjelaskan bahwa perbedaan domain dalam tugas *summarization* dapat menyebabkan model gagal menggeneralisasi dengan baik dan menunjukkan performa yang rendah pada domain tak dikenal [57]. Kedua, model tidak memiliki konteks sebelumnya terkait data baru ini, berbeda dengan data uji yang umumnya lebih mirip dengan data latih [58]. Terakhir, seperti

yang dijelaskan dalam penelitian Silvia Casola dkk. (2025), variasi gaya penulisan ringkasan pada referensi dapat mempengaruhi skor ROUGE, walaupun secara makna ringkasan yang dihasilkan tetap relevan [59]. Penurunan skor ini memberikan gambaran yang lebih realistis mengenai kemampuan generalisasi model ketika diterapkan pada data baru. Hasil skor ROUGE yang didapatkan dari preodiksi model ini terdapat pada Tabel 4.14.

Tabel 4.14 ROUGE *Score* Prediksi Model

ROUGE-1	**ROUGE-2**	**ROUGE-L**
0.48	0.36	0.43

4.8.2 Hasil Ringkasan Prediksi Model

Model BART menunjukkan performa awal yang cukup baik dalam mengenali topik utama dari artikel baru berjudul *"Deep, Deep Learning with BART"*, meskipun tanpa melalui tahap *preprocessing* terlebih dahulu. Ringkasan yang dihasilkan berhasil mencakup poin-poin penting seperti pengembangan *framework* rekonstruksi citra berbasis *deep learning*, pemanfaatan *toolbox* BART, serta integrasi operator nonlinier. Model ini juga berhasil mengekstrak istilah penting seperti *Variational Network* dan *Unrolled Reconstruction*. Namun, ringkasan yang dihasilkan belum sepenuhnya lengkap karena terhenti pada bagian hasil dan mengandung kesalahan penulisan seperti kata *"finetrentiation"* yang kemungkinan disebabkan oleh masalah tokenisasi atau *decoding*, serta tidak menyertakan poin penting pada bagian kesimpulan terkait kontribusi operator nonlinier. Hal ini sejalan dengan karakteristik BART sebagai model dengan tujuan *denoising* yang memungkinkan pembuatan kata baru di luar teks referensinya [25].

Penurunan kualitas ini dapat disebabkan oleh beberapa faktor, salah

satunya adalah batasan panjang *input* maksimum model BART sebanyak 1024 token, yang menyebabkan bagian akhir teks terpotong sebelum diproses [60]. Selain itu, artikel yang diringkas termasuk dalam domain yang berbeda dari data pelatihan model, sehingga model mengalami kesulitan dalam menggeneralisasi struktur dan istilah teknis yang belum pernah ditemui sebelumnya [58]. Struktur kalimat dalam artikel juga cenderung kompleks dan kurang terorganisir dengan baik, yang memengaruhi kemampuan model dalam menangkap informasi utama secara efektif. Seperti yang dijelaskan sebelumnya oleh penelitian Silvia Casola dkk. (2025), yang menekankan bahwa variasi dalam gaya penulisan dan struktur ringkasan referensi dapat berdampak pada performa metrik ringkasan seperti ROUGE, meskipun arti ringkasannya tetap terjaga [59]. Oleh karena itu, meskipun model berhasil menunjukkan kemampuan awal yang cukup baik dalam menangkap inti pada artikel ilmiah, penurunan performa pada beberapa bagian menggambarkan tantangan dalam menangani teks yang panjang dan kompleks. Hasil ringkasan prediksi model tersebut terdapat pada Tabel 4.15.

Tabel 4.15 Hasil Ringkasan Prediksi Model

Judul Artikel	**Teks Referensi**	**Teks Ringkasan BART**
Deep, Deep Learning with BART	*purpose to develop a deep-learning-based image reconstruction framework for reproducible research in mail methods the bart toolbox offers a rich set of implementations of calibration and reconstruction algorithms for parallel imaging and compressed sensing in this work bart was extended by a nonlinear operator framework that provides automatic differentiation to allow computation of gradients existing mri-specific operators of bart such as the nonuniform fast fourier transform are directly integrated into this framework and are complemented by common building blocks used in neural networks to evaluate the use of the framework for advanced deep-learning-based reconstruction two state-of-the-art unrolled reconstruction networks namely the variational network and model were implemented results state-of-the-art deep image-reconstruction networks can be constructed and trained using arts gradient based optimization algorithms the bart implementation achieves a similar performance in terms of training time and reconstruction quality compared to the original implementations based on tensorflow. conclusion by integrating nonlinear operators and neural networks into bart we provide a general framework for deep-learning-based reconstruction in mail.*	*Purpose to develop a deep-learning-based image reconstruction framework for reproducible research in mail methods the bart toolbox offers a rich set of implementations of calibration and reconstruction algorithms for parallel imaging and compressed sensing in this work bart was extended by a nonlinear operator framework that provides automatic finetrentiation to allow computation of gradients existing nonlinear operators of bart such as the nonuniform fast fourier transform are directly integrated into this framework and are complemented by common building blocks used in neural networks to evaluate the use of the framework for advanced deep learning-based reconstruction two state-of-the-art unrolled reconstruction networks namely the variational network and model were implemented results stateoftheart deep image-reconstruction networks.*

BAB V

Penutup

5.1 Kesimpulan

Berdasarkan rumusan masalah dan tujuan yang dikemukakan pada Tugas Akhir ini diperoleh beberapa kesimpulan sebagai berikut:

1. Analisis karakteristik distribusi kata dalam artikel ilmiah bertema *"data science"* dilakukan terhadap dua jenis data, yaitu data mentah hasil *scraping* dari arXiv dan data yang telah melalui tahap *preprocessing*. Tujuan analisis ini adalah untuk memahami pola penggunaan kata dan menilai bagaimana *preprocessing* memengaruhi kualitas teks sebagai *input* untuk model peringkasan. Analisis dilakukan dengan menghitung frekuensi kata, membuat visualisasi *Word Cloud*, serta mempelajari distribusi panjang kalimat. Pada data mentah, kata-kata umum seperti *the, and, of*, serta tanda baca masih mendominasi. Sebaliknya, data yang telah di *preprocessing* memperlihatkan dominasi kata-kata teknis seperti model, data, dan *learning*, dengan penghilangan elemen yang tidak bermakna. *Word Cloud* pada *data preprocessing* juga memperlihatkan visualisasi yang lebih bersih dan rapih. Dari sisi panjang kalimat, hasil *preprocessing* menunjukkan distribusi yang lebih alami dengan mayoritas berada pada rentang 10–40 kata. Perbedaan hasil ini menunjukkan bahwa tahap *preprocessing* berhasil meningkatkan kualitas dan representasi data yang sangat penting dalam mendukung performa model peringkasan teks menggunakan BART.

2. Evaluasi performa model BART dalam peringkasan teks abstraktif dilakukan melalui kombinasi analisis kuantitatif menggunakan ROUGE *Score* dan analisis kualitatif terhadap hasil ringkasan. Berdasarkan percobaan terhadap berbagai kombinasi *hyperparameter*, konfigurasi terbaik diperoleh dengan *learning rate* sebesar 1e-5, *batch size train* 24, *batch size test* 4, *epoch* 5, *weight decay* 0.01, dan *logging steps* 100. Konfigurasi ini menghasilkan skor ROUGE-1 sebesar 0.60, ROUGE-2 sebesar 0.52, dan ROUGE-L sebesar 0.56, yang menunjukkan kesamaan yang cukup tinggi antara ringkasan referensi dengan ringkasan model. Secara kualitatif, ringkasan yang dihasilkan mampu menyampaikan inti informasi dari artikel ilmiah, meskipun masih cenderung terlalu umum dan kurang menggambarkan detail teknis, seperti terlihat pada artikel mengenai pustaka *Python DataSist* maupun metode BiFlag. Pada tahap prediksi menggunakan artikel baru berjudul "*Deep, Deep Learning with BART*", model mampu menangkap bagian awal artikel dengan baik namun kurang mencakup keseluruhan isi, yang mencerminkan keterbatasan dalam menjaga konteks pada teks panjang. Skor ROUGE pada prediksi juga lebih rendah dibanding data uji, yaitu ROUGE-1 sebesar 0.48, ROUGE-2 sebesar 0.36, dan ROUGE-L sebesar 0.43, menunjukkan bahwa performa model menurun ketika dihadapkan pada distribusi data yang berbeda. Secara keseluruhan, BART menunjukkan performa yang cukup baik dalam meringkas teks ilmiah secara abstraktif, namun masih memerlukan peningkatan dalam hal ketepatan informasi teknis dan konteks yang lebih luas.

5.2 Saran

Berdasarkan hasil dan pembahasan dalam penelitian ini, disarankan untuk mengeksplorasi penggunaan model lain seperti T5, PEGASUS, atau *Longformer* sebagai alternatif dalam peringkasan teks. Meskipun evaluasi performa model BART menggunakan ROUGE *Score* menunjukkan hasil yang cukup baik, perbandingan dengan model lain perlu dilakukan untuk mengetahui model yang paling optimal dalam menghasilkan ringkasan teks ilmiah. Selain pengujian kuantitatif seperti ROUGE *Score*, perlu juga dilakukan pengujian kualitatif untuk mengecek *fluency*, *coherence*, dan *relevance* hasil ringkasan model. Terakhir, pengujian model pada artikel ilmiah dari berbagai disiplin ilmu, tidak terbatas pada topik *data science* dapat dilakukan untuk mengukur kemampuan generalisasi model terhadap berbagai jenis konten ilmiah.

DAFTAR PUSTAKA

[1] Elsevier. “Elsevier text and data mining (tdm) license”. [Accessed: 20-Maret-2025]. sumber: https://bit.ly/4j2cW43.

[2] IEEE. “Ieee xplore® api terms of use”. [Accessed: 20-Maret-2025]. sumber: https://bit.ly/3RXMYn7.

[3] I. John Wiley & Sons. “Text and data mining”. [Accessed: 20-Maret-2025]. sumber: https://bit.ly/4jTvogo.

[4] arXiv. “Open-access archive for scholarly articles”. [Accessed: 15-Maret-2025]. sumber: https://bit.ly/43eLFWh.

[5] arXiv. “About arxiv”. [Accessed: 12-Februari-2025]. sumber: https://bit.ly/3GMzici.

[6] I. Atanassova, M. Bertin, dan V. Lariviere, “On the composition of scientific abstracts”, *Journal of Documentation*, vol. 72, no. 4, hlmn. 1–13, 2016.

[7] D. Suleiman dan A. Awajan, “Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges”, *Hindawi : Mathematical Problems in Engineering*, hlmn. 1–29, 2020.

[8] T. Shi, Y. Keneshloo, N. Ramakrishnan, dkk., “Neural abstractive text summarization with sequence-to-sequence models”, *ACM/IMS Transactions on Data Science*, vol. 2, no. 1, hlmn. 1–37, 2020.

[9] I. Cholissodin, A. A. Soebroto, U. Hasanah, dkk., *Buku Ajar AI, Machine Learning & Deep Learning*. ResearchGate, 2019.

[10] G. Hartawan, D. S. Maylawati, dan W. Uriawan, “Bidirectional and auto-regressive transformer(bart) for indonesian abstractive text summarization”, *JIP (Jurnal Informatika Polinema)*, vol. 10, no. 4, hlmn. 535–541, 2024.

[11] N. Shafiq, I. Hamid, M. Asif, Q. Nawaz, H. Aljuaid, dan H. Ali, “Abstractive text summarization of low-resourced

languages using deep learning", *PeerJ Computer Science*, vol. 9, no. 1176, hlmn. 1–22, 2023.

[12] M. O. Topal, A. Bas, dan I. van Heerden, "Exploring transformers in natural language generation: Gpt, bert, and xlnet", *arXiv*, hlmn. 1–3, 2021.

[13] K. Akiyama, A. Tamura, dan T. Ninomiya, "Hie-bart: Document summarization with hierarchical bart", *Association for Computational Linguistics*, hlmn. 159–165, 2021.

[14] M. La Quatra dan L. Cagliero, "Bart-it: An efficient sequence-to-sequence model for italian text summarization", *Future Internet*, vol. 15, no. 15, hlmn. 1–13, 2023.

[15] P. Wilmana, T. Ataraa, dan D. Suhartono, "Abstractive english document summarization using bart model with chunk method", *Procedia Computer Science*, hlmn. 1010–1019, 2024.

[16] Q. A. Itsnaini, M. Hayaty, A. D. Putra, dkk., "Abstractive text summarization using pre-trained language model "text-to-text transfer transformer (t5)"", *ILKOM Jurnal Ilmiah*, vol. 15, no. 1, hlmn. 124–131, 2023.

[17] K. Ivanedra dan M. Mustikasari, "Implementasi metode recurrent neural network pada text summarization dengan teknik abstraktif", *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 6, no. 4, hlmn. 377–382, 2019.

[18] A. R. Lubis, H. R. Safitri, Irvan, dkk., "Enhancing text summarization with a t5 model and bayesian optimization", *Revue d'Intelligence Artificielle*, vol. 37, no. 5, hlmn. 1213–1219, 2023.

[19] M. Kirmani, G. Kaur, dan M. Mohd, "Analysis of abstractive and extractive summarization methods", *International Journal of Emerging Technologies in Learning (iJET)*, vol. 19, no. 1, hlmn. 86–96, 2024.

[20] R. Levin, V. Cherepanova, A. Schwarzschild, dkk., "Transfer learning with deep tabular models", *arXiv*, hlmn. 1–34, 2023.

[21] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images", *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 10, hlmn. 142–150, 2019.

[22] Y. Keneshloo, N. Ramakrishnan, dan C. K. Reddy, "Deep transfer reinforcement learning for text summarization", *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, hlmn. 1–12, 2019.

[23] J. Nagidi. "Transfer learning: Leveraging existing knowledge to enhance your models". [Accessed: 13-Februari-2025]. sumber: https://bit.ly/4dbz4r7.

[24] I. Sutskever, O. Vinyals, dan Q. V. Le, "Sequence to sequence learning with neural networks", *arXiv*, hlmn. 1–9, 2014.

[25] M. Lewis, Y. Liu, N. Goyal, dkk., "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension", *arXiv*, hlmn. 1–10, 2019.

[26] S. Sharma. "Text summarization with bart model". [Accessed: 15-Februari-2025]. sumber: https://bit.ly/3F41yXg.

[27] G. Brauwers dan F. Frasincar, "A general survey on attention mechanisms in deep learning", *arXiv*, vol. 35, no. 4, hlmn. 1–20, 2022.

[28] K. Sun, P. Qi, Y. Zhang, dkk., "Tokenization consistency matters for generative models on extractive nlp tasks", *Findings of the Association for Computational Linguistics*, hlmn. 13 300–13 310, 2023.

[29] R. Sennrich, B. Haddow, dan A. Birch, "Neural machine translation of rare words with subword units", *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, hlmn. 1715–1725, 2016.

[30] Y. Jia, "Attention mechanism in machine translation", *Journal of Physics:Conference Series*, vol. 131, no. 1, hlmn. 1–7, 2019.

[31] A. Vaswani, N. Shazeer, N. Parmar, dkk., "Attention is all you need", *arXiv*, hlmn. 1–15, 2017.

[32] J. Howard dan S. Ruder, "Universal language model fine-tuning for text classification", *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, hlmn. 328–339, 2018.

[33] Z. Wan, Y. Zhang, Y. Wang, dkk., "Reformulating domain adaptation of large language models as adapt-retrieve-revise : A case study on chinese legal domain", *Findings of the Association for Computational Linguistics: ACL*, vol. 1, hlmn. 5030–5041, 2024.

[34] C. Karouzos, G. Paraskevopoulos, dan A. Potamianos, "Udalm: Unsupervised domain adaptation through language modeling", *Association for Computational Linguistics*, vol. 1, hlmn. 2579–2590, 2021.

[35] D. Vucetic, M. Tayaranian, M. Ziaeefard, dkk., "Efficient fine-tuning of bert models on the edge", *arXiv*, vol. 1, hlmn. 1–5, 2022.

[36] A. A. Foysal dan R. Böck, "Who needs external references?text summarization evaluation using original documents", *AI (Switzerland)*, vol. 4, no. 4, hlmn. 970–995, 2023.

[37] C. B. Clement, M. Bierbaum, K. O'Keeffe, dkk., "On the use of arxiv as a dataset", *arXiv*, hlmn. 1–7, 2019.

[38] Kat Boboris. "Arxiv sets new record for monthly submissions". [Accessed: 16-Januari-2025]. sumber: https://bit.ly/4iRMU3f.

[39] M. Thoma. “Extract text from a pdf”. [Accessed: 24-Juli-2025]. sumber: http://bit.ly/3GXZaCv.

[40] M. De Silva. “Preprocessing steps for natural language processing (nlp): A beginner’s guide”. [Accessed: 18-Juli-2025]. sumber: http://bit.ly/3GySScp.

[41] B. Zhao, “Clinical data extraction and normalization of cyrillic electronic health records via deep-learning natural language processing”, *JCO Clinical Cancer Informatics*, hlmn. 1–9, 2019.

[42] P. Martins, F. Cardoso, P. Váz, J. Silva, dan M. Abbasi, “Performance and scalability of data cleaning and preprocessing tools: A benchmark on large real-world datasets”, *Data*, vol. 10, no. 1, hlmn. 1–22, 2025.

[43] E. Mousavi. “Nlp series: Day 2 — text preprocessing and tokenization”. [Accessed: 18-Juli-2025]. sumber: http://bit.ly/4lUUYlK.

[44] F. Alshanik, A. Apon, A. Herzog, dkk., “Accelerating text mining using domain-specific stop word lists”, *arXiv*, hlmn. 1–10, 2020.

[45] nkmk. “Convert strings to lists and dictionaries in python”. [Accessed: 07-Juli-2025]. sumber: http://bit.ly/4lKrFT4.

[46] H. Face, *Transformers: Trainingarguments documentation*, 2023.

[47] M. A. Zamzam, C. Crysdian, dan K. F. H. Holle, “Sistem automatic text summarization menggunakan algoritma textrank”, *Jurnal Ilmu Komputer dan Teknologi Informasi*, vol. 12, no. 2, hlmn. 111–116, 2020.

[48] M. Kunilovskaya dan A. Plum, “Text preprocessing and its implications in a digital humanities project”, di dalam *Proceedings of the Student Research Workshop associated with RANLP-2021*, INCOMA Ltd., 2021, hlmn. 85–93.

[49] M. F. Islam, J. Hasan, M. A. Islam, dkk., "Banglalem: A transformer-based bangla lemmatizer with an enhanced dataset", *Systems and Soft Computing*, vol. 7, no. 1, hlmn. 1–10, 2025.

[50] S. Brightwood, *Exploring descriptive textual data analysis using machine learning techniques*, 2024.

[51] J. Brownlee. "Loss and loss functions for training deep learning neural networks". [Accessed: 16-Juli-2025]. sumber: http://bit.ly/3GNo9YY.

[52] Z. Zhao dan P. Chen, "To adapt or to fine-tune: A case study on abstractive summarization", di dalam *Proceedings of the 21st China National Conference on Computational Linguistics (CCL)*, Technical Committee on Computational Linguistics, Chinese Information Processing Society of China, 2022, hlmn. 824–835.

[53] X. Song, X. Tan, T. Qin, dan T.-Y. Liu, "The bart-based model for scientific articles summarization", *Journal of Universal Computer Science*, vol. 30, no. 13, hlmn. 1808–1828, 2024.

[54] M. Popel dan O. Bojar, "Training tips for the transformer model", *The Prague Bulletin of Mathematical Linguistics*, hlmn. 1–28, 2018.

[55] F. D'Angelo, M. Andriushchenko, A. Varre, dkk., "Why do we need weight decay in modern deep learning?", di dalam *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, Neural Information Processing Systems Foundation, 2024, hlmn. 1–33.

[56] J. Zhang, Y. Zhao, M. Saleh, dkk., "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization", di dalam *Proceedings of the 37th International Conference on Machine Learning (ICML)*, PMLR, 2020, hlmn. 1–12.

[57] D. Wang, P. Liu, M. Zhong, dkk., “Exploring domain shift in extractive text summarization”, *arXiv preprint*, hlmn. 1–11, 2019.

[58] A. Elangovan, K.-W. Chang, I. Beltagy, dkk., “Memorization vs. generalization: Quantifying data leakage in nlp performance evaluation”, di dalam *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2021, hlmn. 2370–2386.

[59] S. Casola, Y. J. Liu, S. Peng, dkk., “Evaluation should not ignore variation: On the impact of reference set choice on summarization metrics”, *arXiv preprint*, hlmn. 1–17, 2025.

[60] W. Chen dan M. Iwaihara, “Efficient summarization of long documents using hybrid extractive-abstractive method”, di dalam *Proceedings of the DEIM Forum 2023*, Waseda University, 2023, hlmn. 1–8.

LAMPIRAN

LAMPIRAN A

Manualisasi BART

1. Data *Scraping*
Contoh data :
judul : *Deep Learning Based Abstractive Text Summarization.*
teks : *This paper reviews deep learning-based abstractive summarization techniques.*

2. *Preprocessing Data*

- *Cleaning Data*
  Menghapus tanda baca dan simbol, mengubah kata menjadi huruf kecil, menghapus tanda baca, serta mengoreksi ejaan kata yang salah. Hasil *cleaning* akan seperti berikut :
  judul : [deep learning based abstractive text summarization]
  teks : [this paper reviews deep learning based abstractive summarization techniques]
- *Tokenization*
  Tokenization akan membagi teks menjadi kata-kata atau unit linguistik yang lebih kecil.
  judul : [’deep’, ’learning’, ’based’, ’abstractive’, ’text’, ’summarization’]
  teks : [’this’, ’paper’, ’reviews’, ’deep’, ’learning’ ’based’, ’abstractive’, ’summarization’, ’techniques’]
- *Stopwords Removal*
  Stopwords adalah kata-kata umum yang dihapus karena tidak memberikan makna signifikan.
  judul : [’deep’, ’learning’, ’based’, ’abstractive’, ’text’, ’summarization’]
  teks : [’paper’, ’reviews’, ’deep’, ’learning’ ’based’,

’abstractive’, ’summarization’, ’techniques’]

- *Lemmatization*
  Lemmatization digunakan untuk mengubah kata-kata tersebut menjadi bentuk dasarnya.
  judul : [’deep’, ’learn’, ’base’, ’abstractive’, ’text’, ’summarization’] teks : [’paper’, ’review’, ’deep’, ’learning’, ’base’, ’abstractive’, ’summarization’, ’technique’]

3. *Tokenizer* Model BART

Pada proses ini, *tokenizer* digunakan untuk mengubah kata-kata dalam teks menjadi numerik agar dapat dipahami oleh model BART. BART menggunakan BPE untuk memproses token katanya. Proses *tokenizer* ini mencakup :

- Inisialisasi *vocabulary* awal
  Pada awalnya, *vocabulary* hanya terdiri dari karakter individu seperti :
  karakter : [d, e, p, l, e, a, r, n, b, s, t, r, c, i, v, u, m, z...]
- Menghitung Frekuensi Pasangan Kata
  Di tahap ini, model akan menghitung pasangan karakter yang paling sering muncul berdampingan seperti :
  ("d, "e") = muncul 1 kali
  ("e", "e") = muncul 2 kali
  ("e", "p") = muncul 1 kali
- Penggabungan Karakter
  Pasangan karakter dengan frekuensi yang lebih banyak akan digabungkan menjadi satu, seperti :
  ("e" + "e") = ("ee")
  Hasil token : [d, ee, p]
- Pembaruan *Vocabulary*
  Setelah semua karakter dihitung frekuensinya dan

digabungkan, kata tersebut akan diberikan token ID unik. Pemberian token ini diberikan berdasarkan frekuensi kemunculan kata tersebut dalam data. Semakin sering kata tersebut muncul, token ID yang diberikan akan semakin kecil, dan sebaliknya. Contoh token ID hasil *tokenizer* adalah sebagai berikut :
judul : [30522, 20310, 2244, 6200, 20190, 9283]

4. Pelatihan Model BART
- *Encoder*

- *Self Attention*
  Self-Attention dalam *Transformer* dihitung dengan menentukan *Query* (*Q*), *Key* (*K*), dan *Value* (*V*), yang diperoleh dari *embedding* token dengan mengalikan matriks bobot (W_Q, W_K, W_V). Nilai bobot didapatkan dari inisialisasi acak. Misalkan bobot transformasi memiliki dimensi 3×3, dengan *embedding* x :

$$X_{\text{deep}} = \begin{bmatrix} 0.2 & 0.8 & 0.6 \end{bmatrix}$$

$$W_Q = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{bmatrix}, \quad W_K = \begin{bmatrix} 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 \\ 0.8 & 0.9 & 1.0 \end{bmatrix},$$

$$W_V = \begin{bmatrix} 0.3 & 0.4 & 0.5 \\ 0.6 & 0.7 & 0.8 \\ 0.9 & 1.0 & 1.1 \end{bmatrix}$$

- Perhitungan *Query* (Q)

$$Q = X.W_Q$$

$$Q_{\text{deep}} = \begin{bmatrix} 0.2 & 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \\ 0.7 & 0.8 & 0.9 \end{bmatrix}$$

$$q_1 = (0.2 \times 0.1) + (0.8 \times 0.4) + (0.6 \times 0.7) = 0.76$$

$$q_2 = (0.2 \times 0.2) + (0.8 \times 0.5) + (0.6 \times 0.8) = 0.92$$

$$q_3 = (0.2 \times 0.3) + (0.8 \times 0.6) + (0.6 \times 0.9) = 1.08$$

$$Q_{\text{deep}} = \begin{bmatrix} 0.76 & 0.92 & 1.08 \end{bmatrix}$$

- Perhitungan *Key* (K)

$$K = X.W_K$$

$$K_{\text{deep}} = \begin{bmatrix} 0.2 & 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 \\ 0.8 & 0.9 & 1.0 \end{bmatrix}$$

$$k_1 = (0.2 \times 0.2) + (0.8 \times 0.5) + (0.6 \times 0.8) = 0.92$$

$$k_2 = (0.2 \times 0.3) + (0.8 \times 0.6) + (0.6 \times 0.9) = 1.08$$

$$k_3 = (0.2 \times 0.4) + (0.8 \times 0.7) + (0.6 \times 1.0) = 1.24$$

$$K_{\text{deep}} = \begin{bmatrix} 0.92 & 1.08 & 1.24 \end{bmatrix}$$

- Perhitungan *Value* (V)

$$V = X.W_V$$

$$V_{\text{deep}} = \begin{bmatrix} 0.2 & 0.8 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.3 & 0.4 & 0.5 \\ 0.6 & 0.7 & 0.8 \\ 0.9 & 1.0 & 1.1 \end{bmatrix}$$

$$v_1 = (0.2 \times 0.3) + (0.8 \times 0.6) + (0.6 \times 0.9) = 1.08$$

$$v_2 = (0.2 \times 0.4) + (0.8 \times 0.7) + (0.6 \times 1.0) = 1.24$$

$$v_3 = (0.2 \times 0.5) + (0.8 \times 0.8) + (0.6 \times 1.1) = 1.40$$

$$V_{\text{deep}} = \begin{bmatrix} 1.08 & 1.24 & 1.40 \end{bmatrix}$$

- *Attention Score*

$$e_{ij} = \frac{Q_i \cdot K_j}{\sqrt{d_k}} \tag{2.2}$$

Misalkan $d_k = 3$:

$$e_{\text{deep}} = \frac{(0.76 \times 0.92) + (0.92 \times 1.08) + (1.08 \times 1.24)}{\sqrt{3}}$$

$$= \frac{0.6992 + 0.9936 + 1.3392}{\sqrt{3}}$$

$$= \frac{3.032}{\sqrt{3}} = 1.75$$

- Softmax untuk Normalisasi

$$\alpha_l = \frac{\exp(e_l)}{\sum_{j=1}^{n} \exp(e_j)} \tag{2.3}$$

Misalkan :

$$e_{deep} = 1.75, \quad e_{learning} = 1.89, \quad e_{base} = 1.65$$

Maka, kita hitung eksponensial dari setiap nilai:

$$\exp(1.75) = 5.754, \quad \exp(1.89) = 6.619, \quad \exp(1.65) = 5.206$$

$$\exp\_total = 5.754 + 6.619 + 5.206 = 17.579$$

Kemudian, kita normalisasi dengan membagi setiap nilai eksponensial dengan total tersebut :

$$\alpha_1 = \frac{5.754}{17.579} = 0.327, \quad \alpha_2 = \frac{6.619}{17.579} = 0.376, \quad \alpha_3 = \frac{5.206}{17.579} = 0.296$$

Jadi, matriks atensi setelah Softmax adalah:

$$\alpha = \begin{bmatrix} 0.327 & 0.376 & 0.296 \end{bmatrix}$$

- *Output Self-Attention*
  Setelah mendapatkan matriks A, hitung output akhir dengan:

$$c = \sum_{l=1}^{n} \alpha_l v_l \tag{2.4}$$

Misalkan matriks *Value* V untuk *deep, learning*, dan *base* adalah:

$$V = \begin{bmatrix} 1.05 & 1.24 & 1.40 \\ 1.08 & 1.20 & 1.42 \\ 1.15 & 1.25 & 1.51 \end{bmatrix}$$

Maka, hasil perkalian:

$$c = \begin{bmatrix} 0.327 & 0.376 & 0.296 \end{bmatrix} \cdot \begin{bmatrix} 1.05 & 1.24 & 1.40 \\ 1.08 & 1.20 & 1.42 \\ 1.15 & 1.25 & 1.51 \end{bmatrix}$$

$$
\begin{aligned}
c_1 &= (0.327 \times 1.05) + (0.376 \times 1.08) + (0.296 \times 1.15) \\
&= 0.3433 + 0.4060 + 0.3404 \\
&= 1.0896
\end{aligned}
$$

$$
\begin{aligned}
c_2 &= (0.327 \times 1.24) + (0.376 \times 1.20) + (0.296 \times 1.25) \\
&= 0.4054 + 0.4512 + 0.37 \\
&= 1.2266
\end{aligned}
$$

$$
\begin{aligned}
c_3 &= (0.327 \times 1.40) + (0.376 \times 1.42) + (0.296 \times 1.51) \\
&= 0.4578 + 0.5339 + 0.4469 \\
&= 2.4194
\end{aligned}
$$

$$
c = \begin{bmatrix} 1.0896 & 1.2266 & 2.4194 \end{bmatrix}
$$

- Decoder

- Input Token *Embedding Decoder*
  Misalkan token *input decoder* saat ini adalah *"deep"* dengan *embedding* :
  $$
  \mathrm{x} = \begin{bmatrix} 0.10 & 0.40 & 0.60 \end{bmatrix}
  $$
- *Context Vector* dari *Encoder*
  $$
  \mathrm{c} = \begin{bmatrix} 1.0896 & 1.2266 & 2.4194 \end{bmatrix}
  $$
- *Masked Self Attention*
  Misal hasil *Multi-Head Attention* untuk *input* **x** dengan *context* **c** adalah :

$$\text{MHA} = \begin{bmatrix} 0.80 & 0.90 & 1.00 \end{bmatrix}$$

- *Add & Norm* 1

  Add : Tambahkan x ke MHA:

$$\begin{aligned} a_1 &= x + \text{MHA} \\ &= \begin{bmatrix} 0.10 + 0.80 & 0.40 + 0.90 & 0.60 + 1.00 \end{bmatrix} \\ &= \begin{bmatrix} 0.90 & 1.30 & 1.60 \end{bmatrix} \end{aligned}$$

  Norm : Hitung *mean* dan standar deviasi dari a_1:

$$\mu = \frac{0.90 + 1.30 + 1.60}{3} = \frac{3.80}{3} = 1.267$$

$$\sigma = \sqrt{\frac{(0.90 - 1.267)^2 + (1.30 - 1.267)^2 + (1.60 - 1.267)^2}{3}} \approx 0.29$$

  Normalisasi :

$$\begin{aligned} n_1 &= \frac{a_1 - \mu}{\sigma} = \begin{bmatrix} \frac{0.90 - 1.267}{0.29} & \frac{1.30 - 1.267}{0.29} & \frac{1.60 - 1.267}{0.29} \end{bmatrix} \\ &\approx \begin{bmatrix} -1.26 & 0.11 & 1.15 \end{bmatrix} \end{aligned}$$

- *Feed Forward Network* (FFN)

  Misal FFN terdiri dari dua lapis linear dan aktivasi ReLU. Bobot FFN :

$$W_1 = \begin{bmatrix} 0.1 & 0.3 & 0.5 \\ 0.2 & 0.4 & 0.3 \\ 0.3 & 0.2 & 0.2 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.2 & 0.1 & 0.4 \\ 0.3 & 0.2 & 0.2 \\ 0.6 & 0.1 & 0.3 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}$$

- Lapisan 1 :

$$z_1 = n_1 W_1 + b_1$$

Hitung :

$$\begin{aligned} z_{1,1} &= (-1.26)(0.1) + (0.11)(0.2) + (1.15)(0.3) + 0.1 \\ &= -0.126 + 0.022 + 0.345 + 0.1 \\ &= 0.341 \end{aligned}$$

$$\begin{aligned} z_{1,2} &= (-1.26)(0.3) + (0.11)(0.4) + (1.15)(0.2) + 0.1 \\ &= -0.378 + 0.044 + 0.23 + 0.1 \\ &= -0.004 \end{aligned}$$

$$\begin{aligned} z_{1,3} &= (-1.26)(0.5) + (0.11)(0.3) + (1.15)(0.2) + 0.1 \\ &= -0.63 + 0.033 + 0.23 + 0.1 \\ &= -0.267 \end{aligned}$$

Aktivasi ReLU (set negatif jadi nol) :

$$a_2 = \text{ReLU}(z_1) = \begin{bmatrix} 0.341 & 0 & 0 \end{bmatrix}$$

- Lapisan 2 :

$$z_2 = a_2 W_2 + b_2$$

Hitung :

$$\begin{aligned} z_{2,1} &= 0.341 \times 0.2 + 0 \times 0.3 + 0 \times 0.6 + 0.1 \\ &= 0.0682 + 0 + 0 + 0.1 = 0.1682 \end{aligned}$$

$$\begin{aligned} z_{2,2} &= 0.341 \times 0.1 + 0 \times 0.2 + 0 \times 0.1 + 0.1 \\ &= 0.0341 + 0 + 0 + 0.1 = 0.1341 \end{aligned}$$

$$\begin{aligned} z_{2,3} &= 0.341 \times 0.4 + 0 \times 0.2 + 0 \times 0.3 + 0.1 \\ &= 0.1364 + 0 + 0 + 0.1 = 0.2364 \end{aligned}$$

- *Add & Norm* 2

  Add : Tambahkan output FFN ke $\mathbf{n_1}$:

$$a_3 = n_1 + z_2 = [-1.26 + 0.1682, \quad 0.11 + 0.1341, \quad 1.15 + 0.2364]$$

$$= \begin{bmatrix} -1.0918 & 0.2441 & 1.3864 \end{bmatrix}$$

  Norm : Hitung *mean* dan standar deviasi :

$$\mu = \frac{-1.0918 + 0.2441 + 1.3864}{3} = \frac{0.5387}{3} = 0.18$$

$$\sigma = \sqrt{\frac{(-1.0918 - 0.18)^2 + (0.2441 - 0.18)^2 + (1.3864 - 0.18)^2}{3}}$$

$$= 1.01$$

Normalisasi :

$$n_2 = \frac{a_3 - \mu}{\sigma} = \left[\frac{-1.0918 - 0.18}{1.01} \quad \frac{0.2441 - 0.18}{1.01} \quad \frac{1.3864 - 0.18}{1.01}\right]$$

$$\approx \left[-1.25 \quad 0.063 \quad 1.194\right]$$

- Proyeksi *Output* ke *Vocabulary* (*Logits*)
  Bobot proyeksi :

$$W_o = \begin{bmatrix} 0.2 & 0.3 & 0.5 & 0.1 & 0.3 & 0.6 \\ 0.1 & 0.4 & 0.3 & 0.2 & 0.2 & 0.1 \\ 0.3 & 0.2 & 0.2 & 0.4 & 0.2 & 0.3 \end{bmatrix}$$

Gabungkan **x** dan *context* **c** :

$$h = [x \parallel c] = \left[0.10 \quad 0.40 \quad 0.60 \quad 1.0896 \quad 1.2266 \quad 2.4194\right]$$

Hitung logit :

$$o_1 = 0.2 \times 0.10 + 0.3 \times 0.40 + 0.5 \times 0.60 + 0.1 \times 1.0896 + 0.3 \times 1.2266 + 0.6 \times 2.4194$$
$$= 0.02 + 0.12 + 0.30 + 0.109 + 0.368 + 1.451 = 2.368$$

$$o_2 = 0.1 \times 0.10 + 0.4 \times 0.40 + 0.3 \times 0.60 + 0.2 \times 1.0896 + 0.2 \times 1.2266 + 0.1 \times 2.4194$$
$$= 0.01 + 0.16 + 0.18 + 0.218 + 0.245 + 0.242 = 1.055$$

$$o_3 = 0.3 \times 0.10 + 0.2 \times 0.40 + 0.2 \times 0.60 + 0.4 \times 1.0896 + 0.2 \times 1.2266 + 0.3 \times 2.4194$$
$$= 0.03 + 0.08 + 0.12 + 0.436 + 0.245 + 0.726 = 1.637$$

- Softmax dalam Prediksi Token

  Hitung eksponen :

$$\begin{aligned} e_{2.368} &\approx 10.68 \\ e_{1.055} &\approx 2.87 \\ e_{1.637} &\approx 5.14 \end{aligned} \quad \Rightarrow \quad \sum = 18.69$$

  Probabilitas token :

$$\begin{aligned} A_1 &= \frac{10.68}{18.69} \approx 0.571 \\ A_2 &= \frac{2.87}{18.69} \approx 0.153 \\ A_3 &= \frac{5.14}{18.69} \approx 0.275 \end{aligned}$$

- Perhitungan *Loss*

$$\mathcal{L}_{DAE}(X, X') = -\sum_{i=1}^{n} \log P(X_i \mid X') \tag{2.1}$$

  Substitusi nilai:

$$\begin{aligned} L_{DAE} &= -(\log(0.571) + \log(0.153) + \log(0.275)) \\ &= -(-0.560 - 1.877 - 1.290) \\ &= 0.560 + 1.877 + 1.290 \\ &= 3.727 \end{aligned}$$

Loss rata-rata per token:

$$L_{DAE} = \frac{3.726}{3} \approx 1.242$$

- Evaluasi dengan *ROUGE Score*
Teks asli : [*This paper reviews deep learning-based abstractive summarization techniques.*]
Teks ringkasan BART : [*A review of deep learning methods for abstractive summarization.*]

Contoh hasil teks setelah di *preprocessing*
Teks asli : ['paper', 'review', 'deep', 'learning', 'base', 'abstractive', 'summarization', 'technique']
Teks ringkasan : ['review', 'deep', 'learning', 'method', 'abstractive', 'summarization']

1. Perhitungan *ROUGE-N*

$$\text{ROUGE-N} = \frac{p}{q} \tag{2.5}$$

- *ROUGE-1*
  Unigram yang sama dari kedua ringkasan :
  ROUGE-1 : (*'review', 'deep', 'learning', 'abstractive', 'summarization'*)

$$ROUGE\text{-}1 = \frac{5}{8} \approx 0.625$$

- *ROUGE-2*
  Bigram yang sama dari kedua ringkasan :
  ROUGE-2 : *('review', 'deep'), ('deep', 'learning'),*

('abstractive', 'summarization')

$$ROUGE\text{-}2 = \frac{3}{7} \approx 0.429$$

2. Perhitungan *ROUGE-L*

$$\text{ROUGE-L} = \frac{\text{LCS}}{m} \tag{2.6}$$

$$ROUGE\text{-}L = \frac{(review,\ deep,\ learning,\ abstractive,\ summarization)}{8}$$

$$= \frac{5}{8} \approx 0.625$$

LAMPIRAN B

Fungsi Program *Scraping* Data

```
import requests
import xml.etree.ElementTree as ET

def scrape_arxiv_by_date(
    query, start_date, end_date,
    total_results=5000, batch_size=100
):
    base_url = "https://export.arxiv.org/api/query"
    all_articles = []

    # Membagi permintaan menjadi batch
    for start in range(0, total_results, batch_size):
        params = {
            "search_query": f"all:{query}",
            "start": start,
            "max_results": batch_size,
            "sortBy": "relevance",
            "sortOrder": "descending"
        }

        # Mengirim permintaan GET
        response = requests.get(base_url, params=params)

        if response.status_code != 200:
            print("Error:", response.status_code); return []


        # Memparsing XML
        root = ET.fromstring(response.text)
        entries = root.findall(
            "{http://www.w3.org/2005/Atom}entry")

        # Mengolah artikel-artikel yang didapatkan
        for entry in entries:
            title = entry.find(
                "{http://www.w3.org/2005/Atom}title"
            ).text.strip()
            summary = entry.find(
                "{http://www.w3.org/2005/Atom}summary"
            ).text.strip()
            published = entry.find(
                "{http://www.w3.org/2005/Atom}published"
            ).text.strip()
            link = entry.find(
                "{http://www.w3.org/2005/Atom}id"
            ).text.strip()
            pdf_link = link.replace("abs", "pdf")

            # Menambahkan informasi author dan
            affiliation
            authors = entry.findall(
                "{http://www.w3.org/2005/Atom}author")
```

```
            author_list = []
            institution_list = []
            for author in authors:
                name = author.find(
                    "{http://www.w3.org/2005/Atom}name")
                if name is not None:
                    author_list.append(name.text.strip())

            # Jika ada informasi institusi
            affiliation = author.find(
                "{http://arxiv.org/schemas/atom}affiliation"
            )
            if affiliation is not None:
                institution_list.append(
                    affiliation.text.strip()
                )

            # Jika tidak ada affiliation yang ditemukan
            if not institution_list:
                institution_list.append("Unknown")

            # Menambahkan artikel ke dalam daftar hasil
            article = {
                "Title": title,
                "Summary": summary,
                "Published": published,
                "Link": link,
                "PDF Link": pdf_link,
                "Authors": ", ".join(author_list)
            }
            all_articles.append(article)

        if len(entries) < batch_size:
            break

    return all_articles
```

LAMPIRAN C

Fungsi Program Ekstraksi *Full Text* PDF

```
def is_valid_url(url):
    try:
        response = requests.head(url, timeout=5)
        return response.status_code == 200
    except requests.exceptions.RequestException:
        return False

# Tambahkan kolom 'Valid URL' ke DataFrame
df['Valid URL'] = df['PDF Link'].apply(is_valid_url)
```

```
# Fungsi untuk mengunduh PDF
def download_pdf(pdf_link, save_path):
    try:
        response = requests.get(pdf_link, timeout=10)
        if response.status_code == 200:
            with open(save_path, 'wb') as file:
                file.write(response.content)
            return True
        else:
            f"Failed to download {pdf_link}: "
            f"{response.status_code}"
            return False
    except requests.exceptions.RequestException as e:
        print(f"Error downloading {pdf_link}: {e}")
        return False
```

```
# Fungsi ekstraksi teks dari PDF
def extract_text_from_pdf(pdf_path):
    try:
        reader = PdfReader(pdf_path)
        full_text = " ".join([
            page.extract_text()
            for page in reader.pages
            if page.extract_text()
        ])
        return full_text
    except Exception as e:
        print(f"Error membaca PDF {pdf_path}: {e}")
        return None
```

```
# Fungsi untuk membersihkan karakter tidak valid
def remove_invalid_characters(text):
    if isinstance(text, str):
        text = re.sub(r'[^\x00-\x7F]+', ' ', text)
        text = re.sub(r'\s+', ' ', text).strip()
    return text
```

LAMPIRAN D

Fungsi Program *Preprocessing* Data

```
# Fungsi cleaning data
def clean_text_for_summarization(text, stopwords_set=None):
    if not isinstance(text, str) or text.strip() == "":
        return "no content"

    # Ubah ke lowercase
    text = text.lower()

    # Hapus semua sebelum dan termasuk kata "abstract"
    text = re.sub(
        r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+',
        '',
        text
    )

    # Hapus email dan URL
    text = re.sub(
        r'[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+',
        '',
        text
    )

    text = re.sub(
        r'http\S+|www\S+|doi:\S+|arxiv:\S+',
        '',
        text
    )

    # Hapus karakter non-ASCII dan simbol aneh
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    text = re.sub(r'[^\w\s.,;:!?()-]', '', text)

    # Hapus kata satu huruf tak bermakna
    text = re.sub(r'\b[b-hj-z]\b', '', text)

    # Hapus spasi berlebih
    text = re.sub(r'\s+', ' ', text).strip()

    return text.strip()
```

```
# Fungsi untuk koreksi ejaan data
def correct_spelling(text):
    words = text.split()
    corrected_words = []

    for word in words:
        suggestions = sym_spell.lookup(word, Verbosity
        .CLOSEST, max_edit_distance=2)
        if suggestions:
            corrected_words.append(suggestions[0].term)
        else:
```

```
            corrected_words.append(word)

    return " ".join(corrected_words)
```

```
# Fungsi untuk normalisasi data
def normalize_tokens(tokens):
    normalized = [
        re.sub(r"[^a-zA-Z.,'\"\s]", "", token).lower()
        for token in tokens
    ]
    return [token for token in normalized if token]
```

LAMPIRAN E

Fungsi Program Pelatihan Model BART

```python
def compute_metrics(eval_pred):
    predictions, labels = eval_pred

    # Pastikan labels dalam bentuk list
    if isinstance(labels, torch.Tensor):
        labels = labels.tolist()

    # BART generate output
    if isinstance(predictions, torch.Tensor):
        predictions = predictions.tolist()
    elif isinstance(predictions, np.ndarray):
        predictions = predictions.tolist()

    labels = [
        [
            token if token != -100 else tokenizer.pad_token_id
            for token in label
        ]
        for label in labels
    ]


    # Decode prediksi dan label
    decoded_preds = tokenizer.batch_decode(
        predictions, skip_special_tokens=True
    )
    decoded_labels = tokenizer.batch_decode(
        labels, skip_special_tokens=True
    )

    # Bersihkan spasi
    decoded_preds = [
        pred.strip() for pred in decoded_preds
    ]
    decoded_labels = [
        label.strip() for label in decoded_labels
    ]

    # Hitung ROUGE
    result = rouge.compute(
        predictions=decoded_preds,
        references=decoded_labels,
        use_stemmer=True
    )

    # F-measure
    result = {
        key: value.mid.fmeasure
        for key, value in result.items()
    }

    return result
```

LAMPIRAN F

Fungsi Program Evaluasi dan Prediksi Model BART

```python
# Fungsi untuk memecah teks panjang menjadi chunk
def split_into_chunks(
    text, tokenizer, max_tokens=512, stride=256
):
    tokens = tokenizer.encode(text, truncation=False)
    chunks = []
    start = 0
    while start < len(tokens):
        end = min(start + max_tokens, len(tokens))
        chunk = tokens[start:end]
        decoded_chunk = tokenizer.decode(
            chunk, skip_special_tokens=True
        )
        chunks.append(decoded_chunk)
        if end == len(tokens):
            break
        start += stride
    return chunks
```

```python
# Fungsi untuk menghasilkan ringkasan dari 1 potongan teks
def bart_summarize(
    input_text, tokenizer, model,
    num_beams=4, num_words=150
):
    try:
        input_text = ' '.join(input_text.split())
        inputs = tokenizer(
            input_text,
            truncation=True,
            max_length=1024,
            return_tensors='pt'
        ).to(model.device)

        summary_ids = model.generate(
            inputs['input_ids'],
            num_beams=num_beams,
            no_repeat_ngram_size=3,
            length_penalty=1.5,
            min_length=30,
            max_length=num_words,
            early_stopping=True
        )

        output = tokenizer.decode(
            summary_ids[0], skip_special_tokens=True
        )
        return clean_summary(output.strip())

    except Exception as e:
        print(f"Error saat merangkum: {e}")
        return "Ringkasan tidak tersedia karena error"
```

```python
# Fungsi untuk meringkas teks menggunakan sliding window
def sliding_summary_with_bart_summarize(
    text, tokenizer, model,
    chunk_size=512, stride=256,
    intermediate_words=100, final_words=120, num_beams=4
):
    chunks = split_into_chunks(
        text, tokenizer, max_tokens=chunk_size,
        stride=stride
    )
    partial_summaries = []

    for chunk in chunks:
        summary = bart_summarize(
            chunk, tokenizer, model,
            num_beams=num_beams,
            num_words=intermediate_words
        )
        partial_summaries.append(summary)

    combined_summary = ' '.join(partial_summaries)
    final_summary = bart_summarize(
        combined_summary, tokenizer, model,
        num_beams=num_beams,
        num_words=final_words
    )
    return final_summary.strip()
```

```python
# Fungsi untuk membuat ringkasan otomatis dan menghitung
skor ROUGE
def generate_and_evaluate_with_rouge(
    test_texts, test_summaries, tokenizer, model, device,
    chunk_size=512, stride=256,
    intermediate_words=100, final_words=120, num_beams=4
):
    generated_summaries = []

    print(
        "Generating summaries with sliding window "
        "+ re-summarization..."
    )
    for text in tqdm(
        test_texts, desc="Generating Summaries"
    ):

        summary = sliding_summary_with_bart_summarize(
            text, tokenizer, model,
            chunk_size=chunk_size,
            stride=stride,
            intermediate_words=intermediate_words,
            final_words=final_words,
            num_beams=num_beams
        )
        generated_summaries.append(summary)

    print("Finished generating. Now computing ROUGE...")

    rouge_scores = rouge.compute(
```

```
        predictions=generated_summaries,
        references=test_summaries
    )

    return generated_summaries, rouge_scores
```